



A Complete Axiomatic System for  
Proving Assertions about  
Recursive and Non-Recursive Programs

Gerald Arthur Gorelick

Technical Report No. 75  
January, 1975

A thesis  
presented to the School of Graduate Studies  
in conformity with the requirements for  
the Degree of Master of Science  
Department of Computer Science  
University of Toronto  
Toronto, Ontario, Canada.

© G. A. Gorelick, 1975.



THE  
LIBRARY OF THE  
UNIVERSITY OF  
MICHIGAN  
ANN ARBOR  
MICHIGAN  
U.S.A.

### Acknowledgments

I am grateful to Professor Stephen Cook for his guidance and the valuable discussions and suggestions which led to the development or improvement of many of the ideas presented in this thesis. I thank Professor Allan Borodin for reading this thesis and suggesting improvements, Jim Donahue for a suggestion which led to a significant simplification of the axiomatic system, and James Allen, upon whose work my proof for the soundness of the rule of recursion is based.

The Department of Computer Science has provided me with an environment most conducive to the pleasurable pursuit of graduate study. For this I am sincerely grateful. I would also like to express my gratitude to the National Research Council for its support.

Indebted as I am to those who have aided me academically, this work is, in other ways, the result of the precious gifts I have received from my parents and sisters, to whom I respond with love and deep gratitude.

1. The first part of the report is devoted to a general  
description of the country and its resources. It  
contains a detailed account of the physical features,  
climate, and population. The second part of the  
report is devoted to a description of the  
economy and the social conditions of the country.  
It contains a detailed account of the principal  
industries, the system of agriculture, and the  
state of the commerce and the arts.

2. The third part of the report is devoted to a  
description of the political system and the  
administration of the country. It contains a  
detailed account of the constitution, the  
legislature, the executive, and the judiciary.  
The fourth part of the report is devoted to a  
description of the military and naval forces of the  
country. It contains a detailed account of the  
army, the navy, and the militia.

3. The fifth part of the report is devoted to a  
description of the education and the public  
works of the country. It contains a detailed  
account of the system of education, the  
public buildings, and the public works.  
The sixth part of the report is devoted to a  
description of the religion and the customs of the  
country. It contains a detailed account of the  
principal religions, the customs, and the  
manners of the people.

### Note to the Reader

Chapter I provides the necessary background to the results which this thesis extends. The reader who is already familiar with Cook [5] will find very little new information here.

Chapter II motivates the modifications which are sufficient for the complete extension of the axiomatic system to recursive programs.

Chapter III presents the proofs of certain lemmas and culminates in the completeness proof for the extended system.

Notation We use Burstall's "cases" notation, where

$f(\alpha) =$

cases  $\alpha$

$\alpha_1 \rightarrow \beta_1$

$\alpha_2 \rightarrow \beta_2$

$\vdots$

indicates that  $f(\alpha)$  is defined as  $\beta_1$  when  $\alpha$  is of form  $\alpha_1$ , etc.

'Cl ( $\alpha$ )' indicates the universal closure of  $\alpha$ , where  $\alpha$  is a wff of some predicate calculus.



## Introduction

We write programs but we are not certain that they do precisely what we intend them to do. We use these programs, and we sometimes discover that, in spite of a convincing performance on trial data sets, a particular program is not entirely "correct": it ends abnormally under some conditions, or it occasionally fails to accomplish what it was devised to accomplish. This disparity between the algorithm that the programmer thinks he has designed and what actually occurs during the execution of his program is a prime motivation for the development of techniques which allow one to prove programs are correct.

The way to prove that a program is correct is to make assertions about what it is doing, and prove that the assertions are in some sense true about the program. The kinds of assertions we allow and which ones we take as true will depend on our assignment of semantics to the program.

The formal deductive theory that is treated in this thesis is based on one proposed by C.A.R. Hoare ([1], [2]), which he derives from work by R.W. Floyd ([3]), for proving the correctness of programs written in a fragment of Algol 60. The definition of semantics of this fragment is a modification of the interpretive model in Lauer [4], as presented in Cook [5]. There are many other approaches to the definition of formal theories and semantics of programs. Among these are the functional approach due to J. McCarthy, D. Scott and



others, R. Burstall's first order logic approach, and the algebraic theory due to S. Igarashi and J.W. de Bakker, as well as other proposals using the interpretive model's constructive approach, differing only in the degree to which it is "machine-oriented". These various directions of research have been discussed and compared in Lauer [4] and Hoare and Lauer [6].

Once we have an interpretive model and a deductive theory for a particular programming language, it is natural to inquire into their relationship. In particular, we are interested in knowing that each program that is correct according to our semantics is provable in our deductive theory, and no incorrect program is provable. These questions of completeness and soundness have been investigated by S.A. Cook ([5]). The Algol fragment that Cook considers is simple but realistically usable. Notably lacking are statement labels and jumps, functions, data structures and recursive procedures. Statement labels, jumps and functions are considered by M. Clint and Hoare ([8]). Work on correctness and data structures has been done by Hoare ([9]) and, more recently, by Cook and D. Oppen ([10], [11]). This thesis addresses the problems involved in extending Cook's work on completeness to an augmented language which includes a rather general capacity for recursion.

## Chapter I

The Algol 60 fragment  $Al[L_1, L_2]$  consists of variable and procedure declarations, and assignment, conditional, while, compound and block statements as in Algol 60, where  $L_1$  is the predicate calculus which supplies the language's variables and expressions, and  $L_2$  is its extension into a language used for assertions.  $L_1$  has a distinguished constant symbol,  $e_0$ , which will stand for the value to which all new variables will be initialized. Procedures are allowed, with restrictions on parameters as follows.

In procedure declarations

$\langle \text{procedure name} \rangle$  proc $(\bar{x}:\bar{v})$   $\langle \text{procedure body} \rangle$ ,

where ' $(\bar{x}:\bar{v})$ ' indicates the formal parameters,  $\bar{x}$  and  $\bar{v}$  are disjoint lists of distinct variables and no variable in  $\bar{v}$  may occur to the left of any assignment statement. In procedure calls

call  $\langle \text{procedure name} \rangle (\bar{a}:\bar{e})$ ,

where ' $(\bar{a}:\bar{e})$ ' indicates the actual parameters,  $\bar{a}$  is a list of distinct variables,  $\bar{e}$  is a list of expressions with no occurrence of a variable in  $\bar{a}$ , and no variable in  $(\bar{a}:\bar{e})$  occurs globally in the procedure body (unless it happens to be a formal parameter).

An interpretive model  $M[I]$  is given by  $(I, R, S, \Delta, \Pi)$ , where  $I$  is an interpretation of  $L_2$ ;  $R$  is an infinite set of

registers  $X_1, X_2, \dots$ ; the set  $S$  of states consists of the mappings  $s : R \rightarrow U$ , where  $U$  is the domain of values given by  $I$ ; the set  $\Delta$  of variable assignments consists of the 1-1 mappings  $\delta : V \rightarrow R$ , where  $V$  is any finite set of variables in  $L_2$ ; and the set  $\Pi$  of procedure assignments consists of the mappings  $\pi : \{\text{procedure names}\} \rightarrow \{\text{procedure bodies}\} \times \{\text{formal parameter lists}\}$ . A formula  $P$  in  $L_2$  is true with respect to a state  $s$  and variable assignment  $\delta$  defined for its free variables  $y_i$  (written:  $\models_{M[I]} P(s, \delta)$ ) iff the formula is true in  $I$  under the interpretation of each  $y_i$  given by the corresponding value  $s(\delta(y_i))$ . (When  $P$  has no free variables, we may write simply  $\models_{M[I]} P$ .) Similarly, an expression  $e$  evaluated with respect to a state  $s$  and a variable assignment  $\delta$  defined for its variables (written  $e(s, \delta)$ ) is the result of applying the expression's operators in the indicated manner to the values of their operands as given by  $s$  and  $\delta$ . Finally, we recursively define the function  $\text{Comp}$ , central to  $M[I]$ , which maps a statement  $A$  and its initial environment, given by  $s$ ,  $\delta$  and  $\pi$ , onto a sequence  $\langle s_1, s_2, \dots \rangle$  of successive states encountered in its computation.

Notation  $A^*$  stands for a sequence  $A_1; A_2; \dots; A_j$ ,  $j \geq 0$ , of statements of  $Al[L_1, L_2]$ .

$D^*$  stands for a sequence  $D_1; D_2; \dots; D_k$ ,  $k \geq 0$ , of declarations of  $Al[L_1, L_2]$ .

$A, A_1 A_2$  each stand for statements of  $Al[L_1, L_2]$ .

$\wedge$  indicates concatenation of sequences of states.

$K \frac{\bar{a}, \bar{e}}{\bar{x}, \bar{v}}$ , where  $K$  is a procedure body, indicates the result of substituting the corresponding actual parameters  $\bar{a}, \bar{e}$  for the free occurrences of the formal parameters  $\bar{x}$  and  $\bar{v}$  (respectively) in  $K$ .

$\text{Out}(A, s, \delta, \pi)$  is the last state in the sequence given by  $\text{Comp}(A, s, \delta, \pi)$ , when this is a finite sequence.

Definition  $\text{Comp}(A, s, \delta, \pi) =$

Cases A:

begin new  $x$ ;  $D^*$ ;  $A^*$  end  $\rightarrow \langle s \rangle^{\wedge} \text{Comp}(\text{begin } D^*; A^* \text{ end}, s', \delta', \pi)$ ,

where  $\delta'(y) = \begin{cases} \delta(y), & \text{if } y \neq x \\ x_{k+1}, & \text{if } y = x, \text{ where } x_{k+1} \text{ is the} \\ & \text{highest-indexed register in the} \\ & \text{range of } \delta, \end{cases}$

and  $s'(x_i) = \begin{cases} s(x_i), & \text{if } x_i \neq \delta'(x) \\ I(e_0), & \text{if } x_i = \delta'(x). \end{cases}$

begin  $p(\bar{x}:\bar{v})$  proc  $K$ ;  $D^*$ ;  $A^*$  end  $\rightarrow \langle s \rangle^{\wedge} \text{Comp}(\text{begin } D^*; A^* \text{ end}, s, \delta, \pi')$ ,

where  $\pi'(q) = \begin{cases} \pi(q), & \text{if } q \neq p \\ \langle K, (\bar{x}:\bar{v}) \rangle, & \text{if } q = p. \end{cases}$

begin  $A_1$ ;  $A^*$  end  $\rightarrow \text{Comp}(A_1, s, \delta, \pi) \wedge \text{Comp}(\text{begin } A^* \text{ end}, \text{Out}(A_1, s, \delta, \pi), \delta, \pi)$ .

begin end  $\rightarrow \langle s \rangle$ .

$x := e \rightarrow \langle s' \rangle$ , where  $s'(x_i) = \begin{cases} s(x_i), & \text{if } \delta(x) \neq x_i \\ e(s, \delta), & \text{if } \delta(x) = x_i \end{cases}$

call  $p(\bar{a}:\bar{e}) \rightarrow \langle s \rangle^{\wedge} \text{Comp}(K \frac{\bar{a}, \bar{e}}{\bar{x}, \bar{v}}, s, \delta, \pi)$ , where  $\pi(p) = \langle K, (\bar{x}:\bar{v}) \rangle$ .

$$\begin{aligned} \underline{\text{if}} \ R \ \underline{\text{then}} \ A_1 \ \underline{\text{else}} \ A_2 &\rightarrow \begin{cases} \langle s \rangle^{\wedge \text{Comp}(A_1, s, \delta, \pi)}, & \text{if } \models_{M[I]} R(s, \delta) \\ \langle s \rangle^{\wedge \text{Comp}(A_2, s, \delta, \pi)}, & \text{otherwise} \end{cases} \\ \underline{\text{while}} \ R \ \underline{\text{do}} \ A_1 &\rightarrow \begin{cases} \text{Comp}(A_1, s, \delta, \pi)^{\wedge \text{Comp}(\underline{\text{while}} \ R \ \underline{\text{do}} \ A_1, \\ \text{Out}(A, s, \delta, \pi), \delta, \pi), \\ \text{if } \models_{M[I]} R(s, \delta) \\ \langle s \rangle, & \text{otherwise} \end{cases} \end{aligned}$$

where  $\delta$  is defined for all variables global in  $A$  and  $\pi$  is defined for all procedure names in  $A$  which occur globally.

The deductive system  $H$  with which we are concerned is Cook's modification ([5]) of the one proposed by Hoare for  $A1[L_1, L_2]$  ([1], [2]). Each formula of  $H$  is either of the form  $P\{A\}Q$ , where  $P, Q$  are formulas of  $L_2$  and  $A$  is a syntactically correct statement of  $A1[L_1, L_2]$ , or it is a formula of  $L_2$ .

Notation  $P, Q, R, S$  stand for formulas of  $L_2$ .

$\frac{\alpha_1, \dots, \alpha_n}{\beta}$  indicates the rule: from the formula(s)  $\alpha_i$ ,  $i=1, \dots, n$  ( $n \geq 1$ ) of  $H$ , deduce the formula  $\beta$  of  $H$ .

$\frac{D, \alpha}{\beta}$ , where  $D$  is a declaration of some procedure  $p$ , indicates the rule given by  $\frac{\alpha}{\beta}$ , with the understanding that all calls of  $p$  in  $\alpha$  and  $\beta$  are in accordance with  $D$ .

The rules and axiom schemata of  $H$  are as follows.

rule of variable declarations

$$\frac{P \frac{y}{x} \ \& \ x = e_0 \{ \underline{\text{begin}} \ D^*; \ A^* \ \underline{\text{end}} \} Q \frac{y}{x}}{P \{ \underline{\text{begin}} \ \underline{\text{new}} \ x; \ D^*; \ A^* \ \underline{\text{end}} \} Q}$$

where y has no occurrence in P, Q, D\* or A\*.

rule of procedure declarations

$$\frac{D_1, \ P \{ \underline{\text{begin}} \ D^*; \ A^* \ \underline{\text{end}} \} Q}{P \{ \underline{\text{begin}} \ D_1; \ D^*; \ A^* \ \underline{\text{end}} \} Q}$$

where D<sub>1</sub> is any procedure declaration.

rule of compound statements

$$\frac{P \{ A \} Q, \ Q \{ \underline{\text{begin}} \ A^* \ \underline{\text{end}} \} R}{P \{ \underline{\text{begin}} \ A; \ A^* \ \underline{\text{end}} \} R}$$

axiom of compound statements

$$P \{ \underline{\text{begin}} \ \underline{\text{end}} \} P$$

axiom of assignment statements

$$P \frac{e}{x} \{ x := e \} P$$

rule of conditional statements

$$\frac{P \& R \{ A_1 \} Q, \ P \& \neg R \{ A_2 \} Q}{P \{ \underline{\text{if}} \ R \ \underline{\text{then}} \ A_1 \ \underline{\text{else}} \ A_2 \} Q}$$

rule of while statements

$$\frac{P \& Q \{ A \} P}{P \{ \underline{\text{while}} \ Q \ \underline{\text{do}} \ A \} P \& \neg Q}$$



rule of non-recursive procedure calls

$$\frac{p(\bar{x}:\bar{v}) \text{ proc } K, P(K)Q}{P(\text{call } p(\bar{x}:\bar{v}))Q}$$

rule of parameter substitution

$$\frac{\bar{v} = \bar{e}\sigma \ \& \ (P\sigma_2) \frac{\bar{x}}{a} \{ \text{call } p(\bar{x}:\bar{v}) \} \bar{v} = \bar{e}\sigma \supset (Q\sigma_2) \frac{\bar{x}}{a}}{P(\text{call } p(a:e))Q}$$

where  $\bar{x}_1 = \bar{x}n\bar{a}$ ,  $\bar{x}_2 = \bar{x} - \bar{x}_1$ ;  $\bar{v}_1 = \bar{v}n\bar{a}$ ,  $\bar{v}_2 = \bar{v} - \bar{v}_1$ ;  $\sigma = \frac{\bar{x}', \bar{v}'}{\bar{x}, \bar{v}}$

where  $\bar{x}', \bar{v}'$  are lists of new variables in 1-1 correspondence to  $\bar{x}, \bar{v}$ ; and  $\sigma_2$  is the restriction of  $\sigma$  to  $\bar{x}_2 \cup \bar{v}_2$ .

Remark This somewhat complex rule is simplified in Chapter II. It is presented here in this form to put later developments into perspective. The reader unfamiliar with this rule can skip its details without affecting his understanding the results of this thesis.

rule of consequence

$$\frac{P \supset R, R(A)S, S \supset Q}{P(A)Q}$$

Remark As they are presented here, the rules of H are less than rigorous in dealing with procedure declarations. If we were only concerned about developing a completely formal deductive system, we would transform each rule  $\frac{\alpha_1, \dots, \alpha_n}{\beta}$  into  $\frac{D^*/\alpha_1, \dots, \alpha_n}{D^*/\beta}$ , explicitly indicating the context of procedure declarations  $D^*$  in all cases. The rule for blocks with

procedure declarations would now become  $\frac{D^*/P\{\underline{\text{begin}}\ D_1^*; A^* \underline{\text{end}}\}Q}{P\{\underline{\text{begin}}\ D^*; D_1^*; A^* \underline{\text{end}}\}Q}$ , enabling us to "discharge" previously implicit assumptions about the context of procedure declarations. Similarly, the rule for procedures would become  $\frac{P\{K\}Q}{p(\bar{x}:\bar{v}) \ \underline{\text{proc}}\ K / P\{\underline{\text{call}}\ p(\bar{x}:\bar{v})\}Q}$ , where  $p$  is a new procedure name,  $\bar{x}$  and  $\bar{v}$  are disjoint lists of distinct variables which occur globally in  $K$ , with no occurrence of any variable in  $\bar{v}$  on the left side of an assignment statement. Now having demonstrated that the informality can be eliminated, we immediately abandon the cumbersome notation such a step would introduce. No confusion should result.

We summarize Cook's results concerning the soundness and completeness of the deductive theory with respect to the interpretive model with the following definitions and theorems. For proofs and more complete explanations, refer to Cook [5].

Definition Given a first order predicate calculus  $L_2$  and an interpretation  $I$  of  $L_2$ , a formula  $P\{A\}Q$  is true in  $M[I]$  (denoted by:  $\models_{M[I]} P\{A\}Q$ ) iff for all states  $s, s'$  such that  $P(s, \delta)$  is true in  $M[I]$  and  $s' = \text{Out}(A, s, \delta, \pi)$ ,  $Q(s', \delta)$  is true in  $M[I]$ , where  $\delta$  is any assignment to the free variables of  $P, Q$  and  $A$ , and  $\pi$  assigns procedure bodies and parameter lists to procedure names in  $A$  according to the context of  $A$ .

Definition Let  $D$  be a proof system for  $L_2$ . A proof in the system  $(H, D)$  is a sequence  $\alpha_1, \dots, \alpha_n$  of formulas  $\alpha$  such that,



for each  $i$ ,  $1 \leq i \leq n$ ,

- i)  $\alpha_i$  is a formula of  $L_2$  and an axiom of  $D$ , or
- ii)  $\alpha_i$  is a formula of form  $P\{A\}Q$  and an axiom of  $H$ , or
- iii)  $\alpha_i$  follows from some  $\alpha_j$ ,  $j < i$ , by a rule of  $D$  or  $H$ .

If  $\alpha$  is a line in some proof in  $(H,D)$  we say  $\alpha$  is provable (denoted by:  $\vdash_{H,D} \alpha$ ). If  $\alpha$  is a line of some proof when  $\beta_1, \dots, \beta_n$  ( $n \geq 1$ ) occur as earlier lines, we write  $\beta \vdash_{H,D} \alpha$ .

Cook, following Lauer, has shown that the axioms are sound for non-recursive programs:

Theorem 1 If  $\vdash_{H,D} P\{A\}Q$  then  $\models_{M[I]} P\{A\}Q$ , where  $I$  is an interpretation of  $L_2$  such that  $D$  is sound relative to  $I$ .

Definition Given the languages  $L_1$  and  $L_2$  and the interpretation  $I$  of  $L_2$  with domain  $U$ , suppose  $A$  is a statement of  $Al[L_1, L_2]$  and  $x_1, \dots, x_n$  are the free variables of  $P$  or  $A$ . Then the post relation corresponding to  $P$  and  $A$  is the relation  $\bar{Q}(x_1, \dots, x_n)$  on  $U$  such that  $\bar{Q}(d_1, \dots, d_n)$  is true iff there is a state  $s$  and variable assignment  $\delta$  such that  $d_i = s'(\delta(x_i))$  for  $i = 1, \dots, n$ , and  $P(s, \delta)$  is true in  $M[I]$ , where  $s' = Out(A, s, \delta, \pi)$ ,  $\delta$  is defined for each  $x_i$ , and  $\pi$  is appropriate to the context of  $A$ . The formula  $Q$  in  $L_2$  expresses the relation  $\bar{Q}$  iff  $Q$  has free variables  $x_1, \dots, x_n$  and  $I(Q_{x_1^1, \dots, x_n^d}) \Leftrightarrow \bar{Q}(d_1, \dots, d_n)$  for all  $d_1, \dots, d_n \in U$ .

Notation  $post(P, A)$  denotes a particular formula in  $L_2$  which expresses the post relation corresponding to  $P$  and  $A$  (say, the one with the smallest Gödel number).

Definition The language  $L_2$  is expressive relative to  $L_1$  and iff

- (i) '=' is in  $L_1$  and receives its standard interpretation in  $I$ , and
- (ii) for every formula  $P$  in  $L_2$  and every statement  $A$  in  $Alt(L_1, L_2)$ ,  $post(P, A)$  is defined.

Theorem 2 (Cook) Let  $T$  be a complete proof system for  $L_2$  (relative to  $I$ ) and suppose  $L_2$  is expressive relative to  $L_1$  and  $I$ . Then  $\models_{M[I]} P\{A\}Q \Rightarrow \vdash_{H, T} P\{A\}Q$ , assuming  $A$  uses no recursion.

## Chapter II

It is only by supplementing the axiomatic system that we can hope to extend the completeness result to recursive programs. If we attempted a proof of some formula about a particular recursive procedure  $p(\bar{x}:\bar{v})$  proc  $K(p)$  (this notation indicates that the procedure body  $K$  contains recursive calls of  $p$ ), the only available means that offers any hope is the rule of non-recursive procedures. Suppose we wanted to prove the formula  $P\{\text{call } p(\bar{x}:\bar{v})\}Q$ . To apply this rule we must satisfy its hypothesis; i.e., we must prove  $P\{K(p)\}Q$ . But in the course of this proof it will be necessary for us to have proven already some formula about the recursive call of  $p$  in the procedure body  $K(p)$ . Since the use of this rule requires that any proof of a formula about a recursive procedure be preceded by a proof of some formula about that procedure, it is inadequate. Hoare solves this problem by introducing a rule that handles the special case of recursive procedures. This rule, as modified by Igarashi, et. al. ([7]) is:

rule of recursive procedures

$$\frac{p(\bar{x}:\bar{v}) \text{ proc } K(p) \quad P\{\text{call } r(\bar{x}:\bar{v})\}Q \vdash P\{K(r)\}Q}{P\{\text{call } p(\bar{x}:\bar{v})\}Q}$$

where  $r$  is a dummy procedure name with no occurrence in  $K(p)$ , and  $K(r)$  stands for  $K$  with all occurrences of  $p$  replaced by  $r$ .



Claim The formula

$$(i) \quad v \geq 0 \{ \text{call fact}(x:v) \} x = v!$$

is provable.

$$(1) \quad 1 = v! \{ x := 1 \} x = v! \quad [\text{axiom of assignment}]$$

$$(2) \quad (v \geq 0 \& v = 0) \supset 1 = v! \quad [\text{theorem of T}]$$

$$(3) \quad x = v! \supset x = v! \quad [\text{theorem of T}]$$

$$(4) \quad v \geq 0 \& v = 0 \{ x := 1 \} x = v! \quad [(1), (2), (3), \text{rule of consequence}]$$

$$(5) \quad x = v! \{ \underline{\text{begin}} \quad \underline{\text{end}} \} x = v! \quad [\text{axiom of compound statements}]$$

$$(6) \quad w \cdot v = v! \{ x = w \cdot v \} x = v! \quad [\text{axiom of assignment}]$$

$$(7) \quad w \cdot v = v! \{ \underline{\text{begin}} \quad x := w \cdot v \quad \underline{\text{end}} \} x = v! \quad [(5), (6), \text{rule of compound statements}]$$

$$(8) \quad w = (v - 1)! \supset w \cdot v = v! \quad [\text{theorem of T}]$$

$$(9) \quad w = (v - 1)! \{ \underline{\text{begin}} \quad x := w \cdot v \quad \underline{\text{end}} \} x = v! \quad [(3), (7), (8), \text{rule of consequence}]$$

(10)  $(v=v'-1 \& v' \geq 0 \& v' \neq 0 \& x=e_0) \supset v \geq 0$  [theorem of T]

(11)  $x=v! \supset (v=v'-1 \supset x=(v'-1)!)$  [theorem of T]

(12)  $v \geq 0 \{ \text{call } r(x:v) \} x=v!$  [assumption]

(13)  $v=v'-1 \& v' \geq 0 \& v' \neq 0 \& x=e_0 \{ \text{call } r(x:v) \} v=v'-1 \supset x=(v'-1)!$   
 [(10), (11), (12), rule  
 of consequence]

(14)  $v \geq 0 \& v \neq 0 \& w=e_0 \{ \text{call } r(w:v-1) \} w=(v-1)!$   
 [(13), rule of parameter  
 substitution]

(15)  $v \geq 0 \& v \neq 0 \& w=e_0 \{ \text{begin call } r(w:v-1); x:=w \cdot v \text{ end} \} x=v!$   
 [(9), (14), rule of  
 compound statements]

(16)  $v \geq 0 \& v \neq 0 \{ \text{begin new } w; \text{call } r(w:v-1); x:=w \cdot v \text{ end} \} x=v!$   
 [(15), rule of variable  
 declarations]

(17)  $v \geq 0 \{ \text{if } v=0 \text{ then } x:=1$   
 $\quad \text{else begin new } w;$   
 $\quad \text{call } r(w:v-1);$   
 $\quad x:=w \cdot v \text{ end} \} x=v!$  [(4), (16), rule of  
 conditional statements]

(18)  $v \geq 0 \{ \text{call fact}(x:v) \} x=v!$  [(1)-(17), rule of  
 recursive procedures]

The fundamental question regarding the completeness of the supplemented deductive theory when recursion is included naturally centers around the new rule. For what formulas (about recursive calls with formal parameters) can the rule's hypothesis be satisfied? We would be fortunate indeed if the proof of every formula went as smoothly as the example above. This is not the case, however. It is easy to see that the formula

$$(ii) \ v \geq 5 \{ \text{call fact}(x:v) \} x=v!$$

is true in our model. In order to prove (ii) using the rule of recursive procedures directly, we would have to use a proof analogous to the one displayed in Example 2.1. At the step corresponding to line (10) the attempt fails, since

$$(v=v'-1 \& v' \geq 5 \& v' \neq 0 \& x=e_0) \supset v \geq 5$$

is false in our model, but it is required in order that the rule of consequence yield the necessary hypothesis to the rule of substitution. It so happens that (ii) is still provable, however, since the rule of consequence allows us to deduce it from (i) and the fact that  $\vdash_T v \geq 5 \supset v \geq 0$ .

Showing completeness for all programs, recursive or not, still would be a simple matter if it could be shown that all true formulas about recursive calls which did not follow from an immediate application of the rule for recursive procedures did follow by the rule of consequence from some formula that already has been proven. Apparently, even this

is not the case. Consider the true formula

(iii)  $v \geq 5 \{ \text{call fact}(x:v) \} x \geq 120$ .

An attempt to prove this by an immediate application of the rule of recursive procedures fails for the same reason that the attempt to use this approach to prove (ii) fails.

Furthermore, we find that the rule of consequence can't be used to derive (iii) from (i) or (ii), since  $x=v! \Rightarrow x \geq 120$  is false in our model.

An approach to a resolution of this difficulty in proving (iii) is indicated by the observation that its truth depends not only on those properties of  $\text{fact}(x:v)$  that make (ii) true, but also on its property that the value of  $v$  remains unchanged when it is invoked. Formula (iii) won't follow from formula (i) because its truth depends on more information than (i) provides. We have this necessary additional information in the formal parameter list for the procedure. Since  $v$  is to the right of the colon, its value cannot change. Perhaps it will suffice to introduce a rule which allows us to glean this sort of information from the syntactic structure of parameter lists. However, with a simple modification we can design a new procedure  $\text{fact1}(x,v:)$ , identical to  $\text{fact}(x:v)$  except for the structure of its parameter list. The value of  $v$  would still remain unchanged by a call to  $\text{fact1}$ , but we wouldn't be permitted to deduce this from the syntax of the call. So our proposed rule is not sufficient and we will need something more powerful.



What we introduce assures that one special, true formula about any particular recursive procedure is directly provable using the rule of recursive procedures, and any other true formula about that procedure follows from this special one. The completeness result for recursive programs is then obtained by exhibiting, for each recursive procedure  $p$ , a "most general formula"  $\alpha_p$  such that  $\vdash \alpha_p$ , and  $\alpha_p \vdash \beta$  for all true formulas  $\beta$  about  $p$ .

Since the formula will be used as the basis for proofs of other true formulas about the procedure, the obvious candidate for such a formula is one that completely indicates the computation performed by any call to the procedure. The following example makes clear just how important it is to record somehow the initial values of variables.

Example 2.2 Let  $L_1 = L_2 = L_N$ , the standard language for number theory. Consider the recursive procedure

```
fact2(x,v:) proc if v = 0 then begin x := 24;
                                     v := 4 end
                                     else call fact2(x,v-1:).
```

We note that

(iv)  $v \geq 0 \{ \text{call fact2}(x,v:) \} x=v!$

is true. The formula

(v)  $v \geq 0 \{ \text{call fact1}(x,v:) \} x=v!$

is also true, as indicated above, but these two procedures are hardly equivalent. It is not simply the case that, similar to the results of a call of fact2, a certain relationship happens to obtain between terminating values of variables after invoking fact1. To show exactly what is computed during a call of fact1, we would have to indicate that the occurrences of 'v' in 'v ≥ 0' and 'x = v!' represent the same value. The technique we use is the introduction of new variables which don't appear in the procedure, permitting ourselves to make use of the obvious fact that their values cannot be changed by any call of the procedure.

Definition Given a procedure  $p(\bar{x}:\bar{v})$  proc K, a most general formula for p is a formula

$$\bar{u} = \bar{c}(\text{call } p(\bar{x}:\bar{v}))\text{post}(\bar{u} = \bar{c}, K),$$

where  $\bar{c}$  is a list of those variables which either are formal parameters or have a global occurrence in K, and  $\bar{u}$  is a list of new variables in  $L_2$  with no occurrence in  $\bar{x}, \bar{v}$ , or K, in 1-1 correspondence with the variables of  $\bar{c}$ .

Remark Every most general formula for any procedure p is true, by the definitions of  $\text{post}(P, A)$  and  $\text{Comp}$ . Note that the only restriction introduced by the precondition  $\bar{u} = \bar{c}$  is that  $\delta$  be defined for the variables of  $\bar{u}$ .

The rules we now adopt are strong enough to allow us to deduce a formula which lacks the new variables  $\bar{u}$  and

yet has a post condition that may depend on the information  $\bar{u}$  provides in the most general formula.

rule of variable substitution

$$\frac{P(\text{call } p(\bar{a}:\bar{e})\}Q}{P\sigma(\text{call } p(\bar{a}:\bar{e})\}Q\sigma},$$

where  $\sigma = \frac{\bar{z}'}{\bar{z}}$  is a substitution of expressions for variables such that i) no variable in  $\bar{z}$  occurs globally in  $K \frac{\bar{a}, \bar{e}}{\bar{x}, \bar{v}}$  (where  $p(\bar{x}:\bar{v}) \text{ proc } K$ ) and ii) for each variable of some expression in  $\bar{z}'$  which happens to occur globally in  $K \frac{\bar{a}, \bar{e}}{\bar{x}, \bar{v}}$ , the corresponding variable(s) of  $\bar{z}$  has no occurrence in  $Q$ .

Remark The substitution  $\sigma$ , like all substitutions in this thesis, is understood to affect only the free occurrences of the indicated variables. Furthermore, it is understood that all bound occurrences of a variable are automatically replaced by a new variable when this is necessary to avoid clashing with the substitution  $\sigma$ .

This rule of variable substitution is sound. The proof of this depends on a general property proved in Cook [5]:

Prop. 1 If  $(s, \delta)$  and  $(s_1, \delta_1)$  are two state, variable assignment pairs such that  $s(\delta(z)) = s_1(\delta_1(z))$  for all  $z$  with a global occurrence in a statement  $A$  of  $Al[L_1, L_2]$ , then, given  $s' = \text{Out}(A, s, \delta, \pi)$  and  $s'_1 = \text{Out}(A, s_1, \delta_1, \pi)$ , either

$s'(\delta(z)) = s'_1(\delta_1(z))$  for all  $z$  global in  $A$  or neither  $s'$  nor  $s'_1$  are defined.

**Lemma 1** For any model  $M[I]$  of  $A_1[L_1, L_2]$ , given a procedure  $p(\bar{x}:\bar{v})$  proc  $K$ , a syntactically correct call call  $p(\bar{a}:\bar{e})$ , and a substitution  $\sigma = \frac{\bar{z}'}{\bar{z}}$  of expressions for variables such that i) no variable in  $\bar{z}$  occurs globally in  $K \frac{\bar{a}, \bar{e}}{\bar{x}, \bar{v}}$  and ii) for each variable of  $\bar{z}'$  which happens to occur globally in  $K \frac{\bar{a}, \bar{e}}{\bar{x}, \bar{v}}$ , the corresponding variable(s) of  $\bar{z}$  has no occurrence in  $Q$ , if  $\models_{M[I]} P\{\text{call } p(\bar{a}:\bar{e})\}Q$ , then  $\models_{M[I]} P\sigma\{\text{call } p(\bar{a}:\bar{e})\}Q\sigma$ .

**Proof** Fix  $M[I]$ . Suppose  $p$  and  $\sigma$  are given as above and  $\models_{M[I]} P\{\text{call } p(\bar{a}:\bar{e})\}Q$ . Suppose that for some state  $s$  and variable assignment  $\delta$ ,  $P\sigma(s, \delta)$  is true in  $M[I]$  and  $s' = \text{Out}(\text{call } p(\bar{a}:\bar{e}), s, \delta, \pi)$  is defined. Define the state  $s_1$ :  $s_1(\delta(\bar{z})) = \bar{z}'(s, \delta)$  and  $s_1(\delta(y)) = s(\delta(y))$  for all  $y$  not in  $\bar{z}$  for which  $\delta(y)$  is defined. Then  $P(s_1, \delta)$  is true in  $M[I]$ .

Since no variable of  $\bar{z}$  has a global occurrence in  $K \frac{\bar{a}, \bar{e}}{\bar{x}, \bar{v}}$ ,  $s(\delta(y)) = s_1(\delta(y))$  for each variable  $y$  with such a global occurrence. By the definition of Comp for procedure calls and by Prop. 1,  $s'_1 = \text{Out}(\text{call } p(\bar{a}:\bar{e}), s_1, \delta, \pi)$  is defined and  $s'(\delta(y)) = s'_1(\delta(y))$  for each  $y$  occurring globally in  $K \frac{\bar{a}, \bar{e}}{\bar{x}, \bar{v}}$ .

Note that  $Q(s'_1, \delta)$  is true in  $M[I]$ . For each variable  $y$  not in  $\bar{z}$  that has no global occurrence in  $K \frac{\bar{a}, \bar{e}}{\bar{x}, \bar{v}}$ ,

$s'_1(\delta(y)) = s_1(\delta(y)) = s(\delta(y)) = s'(\delta(y))$ . Also, for each variable  $z$  in  $\bar{z}$ , if no variable in the corresponding

expression  $z'$  occurs globally in  $K \frac{\bar{a}, \bar{e}}{\bar{x}, \bar{v}}$  then

$s'_1(\delta(z))' = s_1(\delta(z)) = z'(s, \delta) = z'(s', \delta)$ , and if  $z'$  does

include some variable global to  $K \frac{\bar{a}, \bar{e}}{\bar{x}, \bar{v}}$ ,  $z$  doesn't occur in  $Q$ .

From these equalities and the fact that  $\models_{M[I]} Q(s'_1, \delta)$ , it is easy to see that  $Q\sigma(s', \delta)$  is true in  $M[I]$ .

The second new rule which we adopt is:

rule of conjunction

$$\frac{P\{\text{call } p(\bar{a}:\bar{e})\}Q, P\{\text{call } p(\bar{a}:\bar{e})\}S}{P\{\text{call } p(\bar{a}:\bar{e})\}Q\&S},$$

for any assertions  $P, Q, S$  and any procedure  $p$ .

The soundness of this rule is easily established. If  $\models_{M[I]} P\{A\}Q$  and  $\models_{M[I]} P\{A\}S$ , and  $s, \delta$  are such that  $P(s, \delta)$  is true in  $M[I]$  and  $s' = \text{Out}(A, s, \delta, \pi)$  is defined, then  $S(s', \delta)$  is true in  $M[I]$  and  $Q(s', \delta)$  is true in  $M[I]$ , and hence  $Q\&S(s', \delta)$  is true in  $M[I]$ .

Our final addition to the existing system is the following.

axiom of invariance

$$P\{\text{call } p(\bar{a}:\bar{e})\}P,$$

where  $P$  is any assertion which has no variable occurring in  $\bar{a}$  or  $\bar{e}$  (or globally in the procedure body  $K$ , if  $p(\bar{x}:\bar{v}) \text{ proc } K$  has been declared - that is, if  $p$  is not simply a dummy procedure name).

The restriction on the assertion  $P$  guarantees the soundness of this axiom.

Remark If there is no restriction for dummy procedure names corresponding to the axiom's parenthetical restriction for actual procedures, we can deduce some formula  $P\{\text{call } r(\bar{a}:\bar{e})\}P$  with a dummy procedure name  $r$  (corresponding to  $p(\bar{x}:\bar{v}) \text{ proc } K$ ) in which  $P$  has variables global to  $K$ . In particular, if we allow such a deduction in a subproof required by the hypothesis of the rule for recursive procedures, we can prove false formulas. To avoid this difficulty, we may disallow procedure bodies with global variables. There is precedent for this (e.g., Hoare [2]). An alternative is to recognize that this question of soundness with regard to a dummy procedure name only arises when it is associated with an actual procedure name. In such a case, we can apply a restriction similar to that for actual names, requiring that  $P$  not contain any variable global to the body of the actual procedure that is associated with the dummy procedure name.

It is assumed for the remainder of this thesis that one of these alternatives, either restricting the syntax of procedures or restricting the application of this axiom, is chosen. Remaining proofs include the case of variables global to procedure bodies, in case the reader prefers the second choice. Without one of these restrictions the proof for the soundness of the recursion rule will fail.



Remark The rule of conjunction and the axiom of invariance are simple and intuitive. The rule of conjunction is at least derivable for non-recursive procedures  $p$ , by the completeness result for non-recursive programs. The axiom of invariance is derivable for all procedures  $p$ , recursive or non-recursive. The problem with both of these is that there is no apparent way of deriving them when they are applied to dummy procedure names. In such a case we have no procedure body to allow us to use one of the recursion rules, and the other rules which might conceivably apply (the substitution rules and the rule of consequence) seem to be of no help. If there is something that makes proofs of correctness for recursive programs qualitatively more difficult than proofs for non-recursive program, it lies here.

When he first presented his deductive system, C.A.R. Hoare acknowledged the difficulty which prompts our introducing the new rules and axiom. Without making any claims concerning the completeness of the resulting system, he, too, proposes adding a new rule:

rule of adaptation

$$\frac{P\{\underline{\text{call}}\ p(\bar{a}:\bar{e})\}R}{\exists \bar{k}(P \& \forall \bar{a}(R \supset S))\{\underline{\text{call}}\ p(\bar{a}:\bar{e})\}S},$$

where  $\bar{k}$  is a list of all variables free in  $P$  or  $R$  but not in  $\bar{a}, \bar{e}$  or  $S$ . The motivation for this rule is to allow "the assumed properties of a recursive call to be adapted to the

particular circumstances of that call. The formulation of [the] rule of adaptation is designed in such a way so as to permit a mechanically derived answer to the question, 'If S is the desired result of executing a procedure call, call  $p(\bar{a}:\bar{e})$ , and  $P\{\text{call } p(\bar{a}:\bar{e})\}R$  is already given, what is the weakest precondition W such that  $W\{\text{call } p(\bar{a}:\bar{e})\}S$  is universally valid?' (Hoare [2]).

The rule of variable substitution and the rule of conjunction are quite natural and seem much easier to apply than the rule of adaptation, besides being more suited to our approach using most general formulas. In addition to these advantages, the introduction of the rule of variable substitution makes it possible to simplify an already existing rule. The version of the rule of parameter substitution that we have accepted is proposed in Cook [5] as a replacement for the simpler but apparently insufficient rule suggested in Hoare [2]. Hoare's rule is

rule of parameter substitution

$$\frac{P\{\text{call } p(\bar{x}:\bar{v})\}Q}{P\frac{\bar{k}', \bar{a}, \bar{e}}{\bar{k}, \bar{x}, \bar{v}}\{\text{call } p(\bar{a}:\bar{e})\}Q\frac{\bar{k}', \bar{a}, \bar{e}}{\bar{k}, \bar{x}, \bar{v}}},$$

where the substitution  $\frac{\bar{k}'}{\bar{k}}$  indicates the renaming of any variables in P or Q which are not in  $\bar{x}$  or  $\bar{v}$  but which happen to occur in  $\bar{a}$  or  $\bar{e}$ . Cook abandons this simpler rule for the one given in Chapter II in order to guarantee that true



formulas such as  $x=1\{\text{call } p(\bar{a}:)\}x=1$  are provable, where  $p(x:) \text{ proc } K$ . These formulas do seem beyond the power of Hoare's version, but the newly introduced rule of variable substitution complements matters nicely, making Cook's revision unnecessary and allowing us to reinstate Hoare's simpler rule of parameter substitution. Thus, a completeness proof for Hoare's system, even if it included his rule of adaptation, would apparently still require a new rule analogous to the rule of variable substitution. We let  $H'$  denote the deductive system obtained by adding to  $H$  the axiom of invariance and the rules of variable substitution and conjunction, and replacing its rule of parameter substitution with Hoare's rule of parameter substitution. Proofs for the soundness of the latter and for the completeness of  $H'$  for non-recursive programs are found in Appendices I and II.

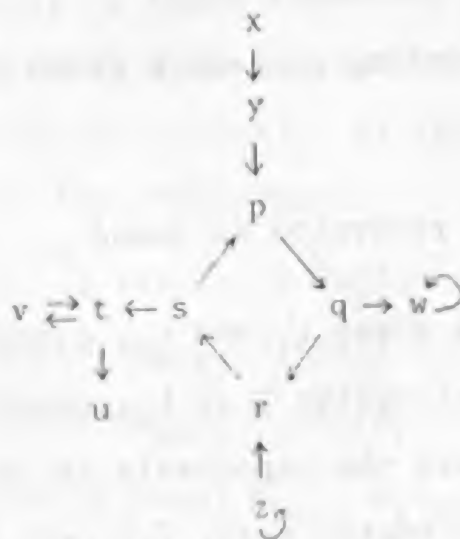
With the rule of variable substitution, we now have a deductive system  $H',T$  that is powerful enough to prove a most general formula  $\alpha$  for any recursive procedure  $p$ , and powerful enough to deduce any true formula about  $p$  from  $\alpha$ . This augmented deductive theory still does not provide us with the means for proving formulas about every sort of program in  $Al[L_1, L_2]$ . In particular, we would like to be able to handle the case where a program  $A$  defines two procedures, each of which calls the other, as well as programs with procedures which are defined in terms of each other in more complicated ways. This requires the introduction of a more general rule,

of which the rule of recursive procedures is simply a special case.

Definition A recursive cycle  $R$  is a set of procedures with the properties:

- a) each procedure in  $R$  calls only procedures in  $R$ , and
- b) there is a non-empty subset  $R'$  of  $R$  such that every procedure in  $R'$  calls some procedure in  $R'$  and is called by some procedure in  $R'$ .

Example 2.4 Procedures are indicated here by letters, and arrows point from invoking procedures to the procedures they invoke.



The following (and only these) are recursive cycles:

$\{w\}$ ,  $\{t, u, v\}$ ,  $\{p, q, r, s, t, u, v, w\}$ ,  
 $\{p, q, r, s, t, u, v, w, z\}$   
 $\{p, q, r, s, t, u, v, w, y\}$   
 $\{p, q, r, s, t, u, v, w, x, y\}$   
 $\{p, q, r, s, t, u, v, w, y, z\}$   
 $\{p, q, r, s, t, u, v, w, x, y, z\}$ .

rule of recursion

$$\begin{array}{c}
 p_i(\bar{x}_i : \bar{v}_i) \text{ proc } A_i(p_1^i, \dots, p_{m_i}^i), \quad i=1, \dots, n \\
 \\
 P_1\{\text{call } r_1(\bar{x}_1 : \bar{v}_1)\}Q_1, \dots, P_n\{\text{call } r_n(\bar{x}_n : \bar{v}_n)\}Q_n \vdash_{H', T} \\
 \\
 P_i\{A_i(r_1^i, \dots, r_{m_i}^i)\}Q_i, \quad 1 \leq i \leq n \\
 \hline
 P_i\{\text{call } p_i(\bar{x}_i : \bar{v}_i)\}Q_i, \quad i=1, \dots, n
 \end{array}$$

where  $p_j^i$  stands for the  $j^{\text{th}}$  procedure to occur (syntactically) in a call in the procedure body  $A_i$  corresponding to procedure  $p_i$ , where  $\{p_i | 1 \leq i \leq n\}$  is a recursive cycle, and where each of the  $n$  distinct dummy procedure names  $r_j^i$  ( $i=1, \dots, n; j=1, \dots, m_i$ ) replaces the corresponding procedure names  $p_j^i$  in the procedure body  $A_i$  of  $p_i$ .

Lemma 2 The rule of recursion is sound.

Proof Suppose we are given  $L_1$  and  $L_2$ , a model  $M[I]$  for  $Al[L_1, L_2]$ , and a proof system  $T$  of  $L_2$ , consistent and complete relative to  $I$ . Suppose the hypothesis of the rule is satisfied by some procedures  $p_i(\bar{x}_i : \bar{v}_i) \text{ proc } A_i(p_1^i, \dots, p_{m_i}^i)$  which form a recursive cycle. By the soundness of the rules and axiom schemata of  $H'$  we know that for each  $j$ ,  $1 \leq j \leq n$ ,  $P_i\{\text{call } r_i(\bar{x}_i : \bar{v}_i)\}Q_i | i=1, \dots, n \vdash_{M[I]} P_j\{A_j(r_1^j, \dots, r_{m_j}^j)\}Q_j$ . (This notation means that when we substitute names of actual procedures for the dummy names, and the formulas on the left

side are true in  $M[I]$ , the formula on the right is true in  $M[I]$ .) Let  $p_j<0>(\bar{x}_j:\bar{v}_j)$  be a procedure which never halts, for  $j = 1, \dots, n$ . Define inductively  $p_j<k>(\bar{x}_j:\bar{v}_j)$  proc  $A_j(p_1^j<k-1>, p_2^j<k-1>, \dots, p_{m_j}^j<k-1>)$  for each  $j, k$ . Note that for each  $k$  and  $j$ ,  $p_j<k>$  behaves exactly like  $p_j$  if there are at most  $k-1$  procedure calls encountered in computing  $\text{Comp}(p_j<k>(\bar{x}_j:\bar{v}_j), s, \delta, \pi)$ , and it never terminates otherwise.

We show that if  $P_i(s, \delta)$  is true in  $M[I]$  and  $s' = \text{Out}(\text{call } p_i(\bar{x}_i:\bar{v}_i), s, \delta, \pi)$ , then  $Q_i(s', \delta)$  is true in  $M[I]$ , for each  $i = 1, \dots, n$ , by induction on the number of procedure calls  $p_j^i$  that are encountered in computing  $\text{Comp}(\text{call } p_i(\bar{x}_i:\bar{v}_i), s, \delta, \pi)$ .

Basis Fix  $i$ . Suppose  $P_i(s, \delta)$  is true in  $M[I]$  and  $s' = \text{Out}(\text{call } p_i<1>(\bar{x}_i:\bar{v}_i), s, \delta, \pi)$ . By the rule's hypothesis and the soundness of the other rules,

$$P_j(\text{call } p_j<0>(\bar{x}_j:\bar{v}_j))Q_j \mid j=1, \dots, n \vdash_{M[I]} P_i(A_i(p_1^i<0>, \dots, p_{m_i}^i<0>))Q_i.$$

Since  $p_j<0>$  never halts,  $\vdash_{M[I]} P_j(\text{call } p_j<0>(\bar{x}_j:\bar{v}_j))Q_j$  for each  $j = 1, \dots, n$ . Hence  $\vdash_{M[I]} P_i(A_i(p_1^i<0>, \dots, p_{m_i}^i<0>))Q_i$  and (using the definition of  $\text{Comp}$ )  $\vdash_{M[I]} P_i(\text{call } p_i<1>(\bar{x}_i:\bar{v}_i))Q_i$ . Thus, by our supposition,  $Q_i(s', \delta)$  is true in  $M[I]$ . Since  $i$  was arbitrary, we have our result for all  $i$ ,  $1 \leq i \leq n$ , when no recursive calls are encountered.

Induction Hypothesis Suppose  $\vdash_{M[I]} P_i(\text{call } p_i<k>(\bar{x}_i:\bar{v}_i))Q_i$  for any  $k \leq K$  and each  $i = 1, \dots, n$ .

Inductive Step Fix  $i$ . Suppose  $P_i(s, \delta)$  is true in  $M[I]$  and  $s' = \text{Out}(\underline{\text{call}} p_i(\bar{x}_i : \bar{v}_i), s, \delta, \pi)$  is defined with  $K$  calls of  $p_j^i$  encountered. Then  $s' = \text{Out}(\underline{\text{call}} p_i^{<K+1>}(\bar{x}_i : \bar{v}_i), s, \delta, \pi)$ . By the hypothesis and the soundness of the other axioms,

$$P_j\{\underline{\text{call}} p_j^{<K>}(\bar{x}_j : \bar{v}_j)\}Q_j \mid j=1, \dots, n \models_{M[I]} P_i\{A_i(p_1^i^{<K>}, \dots, p_{m_i}^i^{<K>})\}Q_i.$$

By the induction hypothesis,  $\models_{M[I]} P_j\{\underline{\text{call}} p_j^{<K>}(\bar{x}_j : \bar{v}_j)\}Q_j$  for each  $j = 1, \dots, n$ . Hence,  $\models_{M[I]} P_i\{A_i(p_1^i^{<K>}, \dots, p_{m_i}^i^{<K>})\}Q_i$ .

But by definition of  $\text{Comp}$ , this means

$\models_{M[I]} P_i\{\underline{\text{call}} p_i^{<K+1>}(\bar{x}_i : \bar{v}_i)\}Q_i$ . Therefore,  $Q_i(s', \delta)$  is true in  $M[I]$ . This completes the induction step, so by mathematical induction  $\models_{M[I]} P_i\{\underline{\text{call}} p_i(\bar{x}_i : \bar{v}_i)\}Q_i$ . Since  $i$  was chosen arbitrarily,  $1 \leq i \leq n$ , this is true for all  $i$ :  $1 \leq i \leq n$ . The rule is therefore sound.

Let  $(H'', T)$  be the deductive system for  $Al[L_1, L_2]$  obtained by adding the rule of recursion to  $(H', T)$ , where  $T$  is a complete proof system for  $L_2$ .

### Chapter III

The proof of the completeness of the system  $(H'', T)$  for all (recursive and non-recursive) programs in  $Al[L_1, L_2]$  depends on a proof that  $\vdash_{M[I]} P\{\underline{\text{call}}\ p(\bar{x}:\bar{v})\}Q \Rightarrow \vdash_{H'', T} P\{\underline{\text{call}}\ p(\bar{x}:\bar{v})\}Q$ , and this is shown by introducing most general formulas, as mentioned earlier. Given a recursive cycle  $R = \{p_1, \dots, p_n\}$ , a most general formula  $P_i\{\underline{\text{call}}\ p_i(\bar{x}_i:\bar{v}_i)\}Q_i$  for each  $p_i$  in  $R$  and any true formula  $\beta$  about these  $p_i$ , we prove that

$$(i) \quad P_i\{\underline{\text{call}}\ r_i(\bar{x}_i:\bar{v}_i)\}Q_i \mid_{i=1, \dots, n} \vdash \beta(r_1, \dots, r_n),$$

where  $\beta(r_1, \dots, r_n)$  indicates the result of substituting a unique dummy procedure name  $r_i$  for each procedure name  $p_i$  in  $\beta$ . Thus, in particular, when we take  $\beta$  to be  $P_j\{A_j\}Q_j$ , where  $A_j$  is the procedure body for  $p_j$ , for some  $j$ ,  $1 \leq j \leq n$ , we have that

$$(ii) \quad \vdash P_j\{\underline{\text{call}}\ p_j(\bar{x}_j:\bar{v}_j)\}Q_j,$$

by the general recursion rule. The proof of (i) will establish that  $P_i\{\underline{\text{call}}\ p_i(\bar{x}_i:\bar{v}_i)\}Q_i \mid_{i=1, \dots, n} \vdash \beta$ . Noting that (ii) holds for any  $j$ ,  $1 \leq j \leq n$ , it will follow that  $\vdash \beta$ .

Definition For any statement  $A$  of  $Al[L_1, L_2]$ , the length of  $A$ , denoted by ' $|A|$ ', is defined recursively as follows:



$|A| =$

cases A:

$x := e \rightarrow 1$

$\underline{\text{begin}} \ \underline{\text{end}} \rightarrow 1$

$\underline{\text{call}} \ p(\bar{a}:\bar{e}) \rightarrow 1$

$\underline{\text{begin}} \ A_1; A^* \ \underline{\text{end}} \rightarrow |A_1| + |\underline{\text{begin}} \ A^* \ \underline{\text{end}}|$

$\underline{\text{begin}} \ \underline{\text{new}} \ x; D^*; A^* \ \underline{\text{end}} \rightarrow |\underline{\text{begin}} \ D^*; A^* \ \underline{\text{end}}| + 1$

$\underline{\text{begin}} \ p(\bar{x}:\bar{v}) \ \underline{\text{proc}} \ K; D^*; A^* \ \underline{\text{end}} \rightarrow |\underline{\text{begin}} \ D^*; A^* \ \underline{\text{end}}| + 1$

$\underline{\text{if}} \ R \ \underline{\text{then}} \ A_1 \ \underline{\text{else}} \ A_2 \rightarrow |A_1| + |A_2| + 1$

$\underline{\text{while}} \ R \ \underline{\text{do}} \ A_1 \rightarrow |A_1| + 1.$

Before beginning the completeness proof we recall some facts proved in Cook [5]:

1.  $\models_{M[I]} P\{\underline{\text{begin}} \ A_1; A^* \ \underline{\text{end}}\}Q \Rightarrow$   
 $\models_{M[I]} P\{A_1\}\text{post}(P, A_1) \ \text{and} \ \models_{M[I]} \text{post}(P, A_1)\{\underline{\text{begin}} \ A^* \ \underline{\text{end}}\}Q.$
2.  $\models_{M[I]} P\{\underline{\text{begin}} \ \underline{\text{new}} \ x; D^*; A^* \ \underline{\text{end}}\}Q \Rightarrow$   
 $\models_{M[I]} P \frac{y}{x} \& x=e_0 \{\underline{\text{begin}} \ D^*; A^* \ \underline{\text{end}}\} Q \frac{y}{x},$  where  $y$  is any new variable with no occurrence in  $P, Q, D^*$  or  $A^*$ .
3.  $\models_{M[I]} P\{\underline{\text{if}} \ R \ \underline{\text{then}} \ A_1 \ \underline{\text{else}} \ A_2\}Q \Rightarrow$   
 $\models_{M[I]} P \& R\{A_1\}Q \ \text{and} \ \models_{M[I]} P \& \neg R\{A_2\}Q.$
4.  $\models_{M[I]} P\{\underline{\text{while}} \ R \ \underline{\text{do}} \ A_1\}Q \Rightarrow$   
 $\models_{M[I]} \text{inv}(P, R, A_1) \& R\{A_1\} \text{inv}(P, R, A_1), \ C1(P \supset \text{inv}(P, R, A_1))$  is valid in  $M[I]$ , and  $C1(\text{inv}(P, R, A_1) \& \neg R \supset Q)$  is valid in  $M[I]$ , where ' $\text{inv}(P, R, A_1)$ ' stands for the loop invariant for this formula as constructed by Cook.

For the remainder of this chapter we shall assume that we have a fixed language  $Al[L_1, L_2]$  and model  $M[I]$  such that  $L_2$  is expressive relative to  $L_1$  and  $I$ , and a fixed proof system  $T$  for  $L_2$ , complete relative to  $I$ .

Lemma 3 Given a procedure  $p(\bar{x}:\bar{v})$  proc  $A$  which may or may not be in a recursive cycle,  $\models_{M[I]} P\{\text{call } p(\bar{a}:\bar{e})\}Q \Rightarrow$   
 $\models_{M[I]} \exists \langle \bar{e} \rangle \tau (\bar{v} = \bar{e}\tau \ \& \ P\tau_{\bar{a}}^{\bar{x}})\{\text{call } p(\bar{x}:\bar{v})\} \exists \langle \bar{e} \rangle \tau (\bar{v} = \bar{e}\tau \ \& \ Q\tau_{\bar{a}}^{\bar{x}}),$

where  $\tau$  indicates the substitution of a unique, fresh variable  $y'$  for each variable  $y$  which is either i) in  $\bar{e}$  or ii) in  $\bar{x}$  or  $\bar{v}$  but not in  $\bar{a}$ , and  $\langle \bar{e} \rangle$  indicates a list of the variables which occur in the expressions  $\bar{e}$ .

Proof Suppose  $\models_{M[I]} P\{\text{call } p(\bar{a}:\bar{e})\}Q$ , and suppose we are given a pair  $s, \delta$  such that  $\exists \langle \bar{e} \rangle \tau (\bar{v} = \bar{e}\tau \ \& \ P\tau_{\bar{a}}^{\bar{x}})(s, \delta)$  is true in  $M[I]$  and  $s' = \text{Out}(\text{call } p(\bar{x}:\bar{v}), s, \delta, \pi)$  is defined. Define the state  $s_1$ :  $s_1(\delta(\bar{a})) = s(\delta(\bar{x}))$ ,  $s_1(\delta(y)) = s(\delta(y'))$  for all variables  $y$  in  $\bar{x}$  or  $\bar{v}$  but not in  $\langle \bar{e} \rangle$  or  $\bar{a}$ ,  $s_1(\delta(\langle \bar{e} \rangle)) = \bar{d}$ , where  $\bar{d}$  are the values which satisfy  $\exists \langle \bar{e} \rangle \tau$  in the precondition, and  $s_1(\delta(y)) = s(\delta(y))$  for all other variables  $y$  for which  $s(\delta(y))$  is defined. Note that  $P(s_1, \delta)$  is true. From the equations  $s_1(\delta(\bar{a})) = s(\delta(\bar{x}))$ ,  $\bar{e}(s_1, \delta) = s(\delta(\bar{v}))$ , and  $s_1(\delta(y)) = s(\delta(y))$  for all variables  $y$  global to  $A$ , it follows by Prop. 1 that  $s'_1 = \text{Out}(\text{call } p(\bar{a}:\bar{e}), s_1, \delta, \pi)$  is defined, with  $s'_1(\delta(\bar{a})) = s'(\delta(\bar{x}))$ ,  $\bar{e}(s'_1, \delta) = \bar{e}(s_1, \delta) = s(\delta(\bar{v})) = s'(\delta(\bar{v}))$ ,  $s'_1(\delta(y)) = s'(\delta(y'))$  for each variable  $y$  in  $\bar{x}$  or  $\bar{v}$  but not in  $\langle \bar{e} \rangle$  or  $\bar{a}$ , and  $s'_1(\delta(y)) = s'(\delta(y))$  for



all other variables  $y$  where  $s'(\delta(y))$  is defined. Hence,  $Q(s'_1, \delta)$  is true in  $M[I]$ . Therefore,  $\exists \langle \bar{e} \rangle \tau (\bar{v} = \bar{e}\tau \ \& \ Q\tau \frac{\bar{x}}{\bar{a}})(s'_1, \delta)$  is true in  $M[I]$ :  $s'_1(\delta(\langle \bar{e} \rangle))$  gives values which satisfy the existential quantifier.

Lemma 4 Given any procedure  $p(\bar{x}:\bar{v})$  proc  $A$ ,

$$\exists \langle \bar{e} \rangle \tau (\bar{v} = \bar{e}\tau \ \& \ P\tau \frac{\bar{x}}{\bar{a}}) \{ \text{call } p(\bar{x}:\bar{v}) \} \exists \langle \bar{e} \rangle \tau (\bar{v} = \bar{e}\tau \ \& \ Q\tau \frac{\bar{x}}{\bar{a}}) \vdash P \{ \text{call } p(\bar{a}:\bar{e}) \} Q,$$

where  $\tau$  indicates the substitution of a unique, fresh variable  $y'$  for each variable  $y$  which is either i) in  $\bar{e}$  or ii) in  $\bar{x}$  or  $\bar{v}$  but not in  $\bar{a}$ , and where  $\langle \bar{e} \rangle$  indicates a list of the variables which occur in the expressions  $\bar{e}$ .

Proof Suppose  $\vdash \exists \langle \bar{e} \rangle \tau (\bar{v} = \bar{e}\tau \ \& \ P\tau \frac{\bar{x}}{\bar{a}}) \{ \text{call } p(\bar{x}:\bar{v}) \} \exists \langle \bar{e} \rangle \tau (\bar{v} = \bar{e}\tau \ \& \ Q\tau \frac{\bar{x}}{\bar{a}})$ .

The deduction goes as follows:

$$\vdash \exists \langle \bar{e} \rangle \tau (\bar{e} = \bar{e}\tau \ \& \ P\tau) \{ \text{call } p(\bar{a}:\bar{e}) \} \exists \langle \bar{e} \rangle \tau (\bar{e} = \bar{e}\tau \ \& \ Q\tau)$$

[parameter substitution]

$$\vdash \exists \langle \bar{e} \rangle \tau (\bar{e} = \bar{e}\tau \ \& \ P\tau|_{\langle \bar{e} \rangle}) \{ \text{call } p(\bar{a}:\bar{e}) \} \exists \langle \bar{e} \rangle \tau (\bar{e} = \bar{e}\tau \ \& \ Q\tau|_{\langle \bar{e} \rangle})$$

[variable substitution]

( $\tau|_{\langle \bar{e} \rangle}$  indicates the restriction of the substitution  $\tau$  to variables in  $\langle \bar{e} \rangle$ .) Noting that  $\vdash P \Rightarrow \exists \langle \bar{e} \rangle \tau (\bar{e} = \bar{e}\tau \ \& \ P\tau|_{\langle \bar{e} \rangle})$  (the corresponding values of  $\langle \bar{e} \rangle$  satisfy the quantifier) and

$\vdash \exists \langle \bar{e} \rangle \tau (\bar{e} = \bar{e}\tau \ \& \ Q\tau|_{\langle \bar{e} \rangle}) \Rightarrow Q$ , we have  $\vdash P \{ \text{call } p(\bar{a}:\bar{e}) \} Q$  by the rule of consequence.

Lemma 5 Given any procedure  $p(\bar{x}:\bar{v})$  proc  $A$ , let  $\bar{c}$  be a list of all variables either in  $\bar{x}$  or  $\bar{v}$  or with a global occurrence in  $A$ . If  $\vdash_{M[I]} R \{ \text{call } p(\bar{x}:\bar{v}) \} S$  for some assertions  $R$  and  $S$

in  $L_2$ , then  $\bar{u} = \bar{c}\{\text{call } r(\bar{x}:\bar{v})\}\text{post}(\bar{u}=\bar{c}, A) \vdash_{H', T} R\{\text{call } r(\bar{x}:\bar{v})\}S$ , where  $\bar{u}$  is any list of new variables (i.e. not in  $R, S, \bar{c}$ ) in 1-1 correspondence with those in  $\bar{c}$ , and  $r$  is any procedure name such that the indicated call is syntactically correct.

Proof Suppose  $p(\bar{x}:\bar{v})$ ,  $\bar{u}$  and  $\bar{c}$  are given as above, with  $\vdash_{M[I]} R\{\text{call } p(\bar{x}:\bar{v})\}S$  and suppose  $\vdash_{H', T} \bar{u} = \bar{c}\{\text{call } r(\bar{x}:\bar{v})\}\text{post}(\bar{u}=\bar{c}, A)$ . We must show  $\vdash_{H', T} R\{\text{call } r(\bar{x}:\bar{v})\}S$ . By the axiom of invariance,  $\vdash R_{\bar{c}}^{\bar{u}}\{\text{call } r(\bar{x}:\bar{v})\}R_{\bar{c}}^{\bar{u}}$ . Therefore, by the rule of conjunction,  $\vdash R_{\bar{c}}^{\bar{u}}\bar{u} = \bar{c}\{\text{call } r(\bar{x}:\bar{v})\}R_{\bar{c}}^{\bar{u}}\text{post}(\bar{u}=\bar{c}, A)$ . Since  $R\bar{u} = \bar{c} \supset R_{\bar{c}}^{\bar{u}}\bar{u} = \bar{c}$ ,  $\vdash R\bar{u} = \bar{c}\{\text{call } r(\bar{x}:\bar{v})\}R_{\bar{c}}^{\bar{u}}\text{post}(\bar{u}=\bar{c}, A)$ . Now, suppose  $\text{post}(\bar{u}=\bar{c}, A) \& R_{\bar{c}}^{\bar{u}}(s', \delta)$  is true in  $M[I]$  for some  $s', \delta$  pair. Then there is a state  $s$  such that  $\bar{u} = \bar{c}(s, \delta)$  is true in  $M[I]$  and  $s' = \text{Out}(A, s, \delta, \pi)$ . Let  $\bar{z}$  be a list of those variables in  $R$  or  $S$  which aren't in  $\bar{c}$ . Since no variable in  $\bar{u}$  or  $\bar{z}$  occurs globally or as a parameter in  $A$ ,  $s'(\delta(\bar{u})) = s(\delta(\bar{u}))$  and  $s'(\delta(\bar{z})) = s(\delta(\bar{z}))$ . So  $R_{\bar{c}}^{\bar{u}}(s, \delta)$  is true in  $M[I]$ . Since  $\bar{u} = \bar{c}(s, \delta)$  is true,  $R(s, \delta)$  is true in  $M[I]$ . Since  $\text{Out}(\text{call } p(\bar{x}:\bar{v}), s, \delta, \pi) = \text{Out}(A, s, \delta, \pi)$  and it is assumed that  $\vdash_{M[I]} R\{\text{call } p(\bar{x}:\bar{v})\}S$ ,  $S(s', \delta)$  is true in  $M[I]$ .

Having thus shown that  $\text{post}(\bar{u}=\bar{c}, A) \& R_{\bar{c}}^{\bar{u}} \supset S$  is valid in  $M[I]$ , by the completeness of  $T$  and the rule of consequence we have  $\vdash R\bar{u} = \bar{c}\{\text{call } r(\bar{x}:\bar{v})\}S$ . Applying the rules of variable substitution and consequence,  $\vdash_{H', T} R\{\text{call } r(\bar{x}:\bar{v})\}S$ .

Corollary If  $\models_{M[1]} R(\text{call } p(\bar{x}:\bar{v}))S$ , then

$\bar{u}=\bar{c}(\text{call } p(\bar{x}:\bar{v}))\text{post}(\bar{u}=\bar{c},A) \vdash_{H,T} R(\text{call } p(\bar{x}:\bar{v}))S$ , where all notation is as given in the above lemma.

Lemma 6 Given a statement  $A(p_1, \dots, p_n)$  of  $A_1[L_1, L_2]$ , where  $\text{call } p_j(\bar{a}_j:\bar{c}_j)$  is the  $i^{\text{th}}$  procedure call (syntactically) in  $A$  of any procedure in any recursive cycle, if  $\models_{M[1]} P(A(p_1, \dots, p_n))Q$  then  $\bar{u}_j=\bar{c}_j(\text{call } r_j(\bar{x}_j:\bar{v}_j))\text{post}(\bar{u}_j=\bar{c}_j, A_j) \mid_{j=1, \dots, n} \vdash P(A(r_1, \dots, r_n))Q$ , where the  $r_j$  are any syntactically correct procedure names,  $A(r_1, \dots, r_n)$  is  $A(p_1, \dots, p_n)$  with  $p_j$  replaced by  $r_j$  for  $j = 1, \dots, n$ , and, for each  $j = 1, \dots, n$ ,  $\bar{c}_j$  is a list of the variables global to the procedure body  $A_j$  of  $p_j$  (including  $\bar{x}_j$  and  $\bar{v}_j$ ) and  $\bar{u}_j$  is a list of completely new variables in 1-1 correspondence with those in  $\bar{c}_j$ . The lists  $\bar{u}_j$  have the property that the two variables in  $\bar{u}_j$  or  $\bar{u}_k$  corresponding to a variable  $c$  common to  $\bar{c}_j$  and  $\bar{c}_k$  are identical.

Proof (by induction on  $|A|$ ).

Basis  $|A| = 1$ .

case i  $A$  is of form  $x := e$ . Then  $A(r_1, \dots, r_n)$  is  $A$  and the result follows by the completeness result for non-recursive programs.

case ii  $A$  is of form  $\text{call } p_1(\bar{a}:\bar{c})$ . Suppose  $\models_{M[1]} P(\text{call } p_1(\bar{a}:\bar{c}))Q$ . Then by lemma 3,  
 $\models_{M[1]} \exists \langle \bar{c} \rangle \tau (\bar{v}=\bar{c}\tau \ \& \ P\tau \frac{\bar{x}}{a})(\text{call } p_1(\bar{x}:\bar{v})) \exists \langle \bar{c} \rangle \tau (\bar{v}=\bar{c}\tau \ \& \ Q\tau \frac{\bar{x}}{a}),$

where  $\langle \bar{e} \rangle$  and  $\tau$  are as indicated in the lemma. So, by lemma 5,  $\bar{u}_1 = \bar{c}_1 \{ \underline{\text{call}} \ r_1(\bar{x}:\bar{v}) \} \text{post}(\bar{u}_1 = \bar{c}_1, A_1) \vdash$

$\exists \langle \bar{e} \rangle \tau \ (\bar{v} = \bar{e} \tau \ \& \ P \tau \frac{\bar{x}}{a}) \{ \underline{\text{call}} \ r_1(\bar{x}:\bar{v}) \} \exists \langle \bar{e} \rangle \tau \ (\bar{v} = \bar{e} \tau \ \& \ Q \tau \frac{\bar{x}}{a})$ . Noting

that lemma 4 is a purely syntactic property (i.e. its proof is still valid if we substitute any other syntactically correct procedure name for  $p$ ), we have

$\bar{u}_1 = \bar{c}_1 \{ \underline{\text{call}} \ p_1(\bar{x}:\bar{v}) \} \text{post}(\bar{u}_1 = \bar{c}_1, A_1) \vdash P \{ \underline{\text{call}} \ p_1(\bar{a}:\bar{e}) \} Q$ .

Induction Hypothesis We assume the lemma is true for all statements  $A$  with  $|A| \leq k$ .

Inductive Step Let  $A$  be a statement with  $|A| = k+1$ .

case i  $A$  is of form if  $R$  then  $B_1(p_1, \dots, p_m)$  else

$B_2(p_{m+1}, \dots, p_n)$ . If  $\models_{M[I]} P(A)Q$ , then  $\models_{M[I]} P \& R \{ B_1(p_1, \dots, p_m) \} Q$  and  $\models_{M[I]} P \& \neg R \{ B_2(p_{m+1}, \dots, p_n) \} Q$ . The induction hypothesis applies to each of these, so

$\bar{u}_j = \bar{c}_j \{ \underline{\text{call}} \ r_j(\bar{x}_j:\bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j) \mid j=1, \dots, m \vdash P \& R \{ B_1(r_1, \dots, r_m) \} Q$   
and  $\bar{u}_j = \bar{c}_j \{ \underline{\text{call}} \ r_j(\bar{x}_j:\bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j) \mid j=m+1, \dots, n \vdash$   
 $P \& \neg R \{ B_2(r_{m+1}, \dots, r_n) \} Q$ . Using the rule of conditional statements,  $\bar{u}_j = \bar{c}_j \{ \underline{\text{call}} \ r_j(\bar{x}_j:\bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j) \mid j=1, \dots, n \vdash$   
 $P(A(r_1, \dots, r_n))Q$ .

case ii  $A$  is of form while  $R$  do  $B_1$ . If  $\models_{M[I]} P(A)Q$ , then  $\models_{M[I]} \text{inv}(P, R, A_1) \& R(A_1(p_1, \dots, p_n)) \text{inv}(P, R, A_1)$ . The induction hypothesis applies here and allows us to conclude that

$\bar{u}_j = \bar{c}_j \{ \underline{\text{call}} \ r_j(\bar{x}_j:\bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j) \mid j=1, \dots, n \vdash$   
 $\text{inv}(P, R, B_1) \& R \{ B_1(r_1, \dots, r_n) \} \text{inv}(P, R, B_1)$ . Using the rule of

while statements,  $\bar{u}_j = \bar{c}_j \{ \text{call } r_j(\bar{x}_j : \bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j) \mid j=1, \dots, n \vdash$   
 $\text{inv}(P, R, B_1) \{ B_1(r_1, \dots, r_n) \} \text{inv}(P, R, B_1) \& \neg R$ . Since

$\text{Cl}(P \supset \text{inv}(P, R, B_1))$  and  $\text{Cl}(\text{inv}(P, R, B_1) \& \neg R \supset Q)$  are valid, by the completeness of T and the rule of consequence we have

$\bar{u}_j = \bar{c}_j \{ \text{call } r_j(\bar{x}_j : \bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j) \mid j=1, \dots, n \vdash$   
 $P\{B_1(r_1, \dots, r_n)\}Q$ .

case iii A is of form begin  $B_1(p_1, \dots, p_m); B^*(p_{m+1}, \dots, p_n)$   
end. If  $\models_{M[I]} P\{A\}Q$ , then  $\models_{M[I]} P\{B_1\}\text{post}(P, B_1(p_1, \dots, p_m))$   
and  $\models_{M[I]} \text{post}(P, B_1(p_1, \dots, p_m))\{\text{begin } B^* \text{ end}\}Q$ . The induction hypothesis applies, giving us

$\bar{u}_j = \bar{c}_j \{ \text{call } r_j(\bar{x}_j : \bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j) \mid j=1, \dots, m \vdash$   
 $P\{B_1(r_1, \dots, r_m)\}\text{post}(P, B_1)$  and

$\bar{u}_j = \bar{c}_j \{ \text{call } r_j(\bar{x}_j : \bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j) \mid j=m+1, \dots, n \vdash$   
 $\text{post}(P, B_1)\{\text{begin } B^*(r_{m+1}, \dots, r_n) \text{ end}\}Q$ . Thus, by the rule of compound statements,  $\bar{u}_j = \bar{c}_j \{ r_j(\bar{x}_j : \bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j) \mid j=1, \dots, n \vdash$   
 $P\{A(r_1, \dots, r_n)\}Q$ .

case iv A is of form begin new x;  $D^*$ ;  $A^*$  end. If  $\models_{M[I]} P\{A\}Q$ , then  $\models_{M[I]} P \frac{Y}{x} \& x = e_0 \{ \text{begin } D^*; A^* \text{ end} \} Q \frac{Y}{x}$ , where y is a new variable with no occurrence in P, Q,  $D^*$  or  $A^*$ . The induction hypothesis applies and so

$\bar{u}_j = \bar{c}_j \{ \text{call } r_j(\bar{x}_j : \bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j) \mid j=1, \dots, n \vdash$   
 $P \frac{Y}{x} \& x = e_0 \{ \text{begin } D^*; A^*(r_1, \dots, r_n) \text{ end} \} Q \frac{Y}{x}$ . Applying the rule of variable declarations we have

$\bar{u}_j = \bar{c}_j \{ \text{call } r_j(\bar{x}_j : \bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j) \mid j=1, \dots, n \vdash P\{A(r_1, \dots, r_n)\}Q$ .

These four are only possible cases, so, by induction, the lemma is proved.



Remark The procedure names  $r_i$  are introduced to make it clear that the indicated deduction is purely syntactic - in practice, we will use this result with the actual procedure names  $p_i$  (in which case  $A(r_1, \dots, r_n)$  is simply  $A$ ) or with dummy procedure names required by the general recursion rule.

Theorem 3 Given any statement  $A$  of  $Al[L_1, L_2]$ , if  $\models_{M[I]} P\{A\}Q$  then  $\vdash_{H'', T} P\{A\}Q$ , where  $P$  and  $Q$  are any assertions in  $L_2$ .

Proof If  $A$  is a non-recursive program, the result follows by Theorem 2 and the arguments in Appendix II. On the other hand, suppose  $A$  invokes some procedure in a recursive cycle. Let  $p_1, \dots, p_n$  be the syntactically ordered list of all such procedures in  $A$ . (Note that this is not necessarily a list of distinct procedures.) Suppose  $\models_{M[I]} P\{A\}Q$  for some assertions  $P, Q$  of  $L_2$ . Then, by lemma 6,

$\bar{u}_j = \bar{c}_j \{ \text{call } p_j(\bar{x}_j : \bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j) \mid j=1, \dots, n \vdash P\{A\}Q$ , where the lists of global variables  $\bar{c}_j$  and new variables  $\bar{u}_j$  and the dummy procedure names  $r_j$  are as indicated in that lemma.

Noting that  $\models_{M[I]} \bar{u}_j = \bar{c}_j \{ A_j \} \text{post}(\bar{u}_j = \bar{c}_j, A_j)$  for each  $j$ ,  $1 \leq j \leq n$ , by lemma 6 we also have that

$\bar{u}_k = \bar{c}_k \{ \text{call } r_k(\bar{x}_k : \bar{v}_k) \} \text{post}(\bar{u}_k = \bar{c}_k, A_k) \mid k=1, \dots, n \vdash$

$\bar{u}_j = \bar{c}_j \{ A_j(r_1^j, \dots, r_{m_j}^j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j)$  for each  $j$ ,  $1 \leq j \leq n$ . This

is the hypothesis of the general recursion rule, so we have that  $\vdash \bar{u}_j = \bar{c}_j \{ \text{call } p_j(\bar{x}_j : \bar{v}_j) \} \text{post}(\bar{u}_j = \bar{c}_j, A_j)$  for each  $j$ ,  $1 \leq j \leq n$ .

Thus,  $\vdash_{H'', T} P\{A\}Q$ .

Appendix I. Soundness of Hoare's parameter substitution rule

Suppose  $\models_{M[I]} P\{\underline{\text{call}}\ p(\bar{x}:\bar{v})\}R$  for some procedure  $p(\bar{x}:\bar{v})$  proc  $\Lambda$ . Then  $\models_{M[I]} P\frac{\bar{k}', \bar{a}, \bar{e}}{\bar{k}, \bar{x}, \bar{v}}\{\underline{\text{call}}\ p(\bar{a}:\bar{e})\}R\frac{\bar{k}', \bar{a}, \bar{e}}{\bar{k}, \bar{x}, \bar{v}}$ , where  $\bar{k}$  is a list of those variables not in  $\bar{x}$  or  $\bar{v}$  which occur in  $\bar{a}$  or in some expression in  $\bar{e}$ , and  $\bar{k}'$  is a list of new variables in 1-1 correspondence with  $\bar{k}$ .

Proof Given  $\models_{M[I]} P\{\underline{\text{call}}\ p(\bar{x}:\bar{v})\}R$ , suppose  $P\frac{\bar{k}', \bar{a}, \bar{e}}{\bar{k}, \bar{x}, \bar{v}}(s, \delta)$  is true in  $M[I]$  and  $s' = \text{Out}(\underline{\text{call}}\ p(\bar{a}:\bar{e}), s, \delta, \pi)$  is defined, for some state  $s$  and variable assignment  $\delta$ . Define the state  $s_1$ :  $s_1(\delta(\bar{x})) = s(\delta(\bar{a}))$ ,  $s_1(\delta(\bar{v})) = \bar{e}(s, \delta)$ ,  $s_1(\delta(\bar{k})) = s(\delta(\bar{k}'))$ ,  $s_1(\delta(y)) = s(\delta(y))$  for each  $y$  not in  $\bar{x}, \bar{v}$  or  $\bar{k}$  for which  $s(\delta(y))$  is defined. Since  $\bar{x}, \bar{v}, \bar{k}$  are disjoint lists, this is well-defined. Note that  $P(s_1, \delta)$  is true in  $M[I]$ . By Prop. 1 and the definition of Comp for procedure calls,  $s'_1 = \text{Out}(\underline{\text{call}}\ p(\bar{x}:\bar{v}), s_1, \delta, \pi)$  is defined, with  $s'_1(\delta(\bar{x})) = s'(\delta(\bar{a}))$ ,  $s'_1(\delta(\bar{v})) = \bar{e}(s, \delta) = \bar{e}(s', \delta)$ ,  $s'(\delta(\bar{k}')) = s(\delta(\bar{k}'))$ ,  $s'_1(\delta(\bar{k})) = s_1(\delta(\bar{k}))$  (because we disallow occurrences of  $\bar{a}$  in the procedure body), and  $s'_1(\delta(y)) = s'(\delta(y))$  for all other variables  $y$  where this is defined. Therefore,  $s'_1(\delta(\bar{k})) = s'(\delta(\bar{k}'))$ .  $\models_{M[I]} R(s'_1, \delta)$ , so  $\models_{M[I]} R\frac{\bar{k}', \bar{a}, \bar{e}}{\bar{k}, \bar{x}, \bar{v}}(s', \delta)$ .



Appendix II. Completeness of  $(H', T)$  for non-recursive programs

The proof is identical to Cook's proof for the completeness of II, except for the case of procedure calls in the inductive step. We must show that  $\models_{M[I]} P\{\underline{\text{call}} p(\bar{a}:\bar{e})\}Q \Rightarrow \vdash P\{\underline{\text{call}} p(\bar{a}:\bar{e})\}Q$ , where  $\|\underline{\text{call}} p(\bar{a}:\bar{e})\| = k+1$  and the induction hypothesis states that the axioms are complete for all programs  $B$  with  $\|B\| \leq k$ . (The notation ' $\|B\|$ ' stands for the number of procedure calls in  $B$  plus the length of the result of substituting the corresponding procedure body (with formal parameters replaced by actual parameters) for each procedure call.) If  $\models_{M[I]} P\{\underline{\text{call}} p(\bar{a}:\bar{e})\}Q$ , then by lemma 6  $\models_{M[I]} \exists \langle \bar{e} \rangle_{\tau} (\bar{v} = \bar{e}\tau \ \& \ P\tau_{\frac{\bar{x}}{a}})\{\underline{\text{call}} p(\bar{x}:\bar{v})\} \exists \langle \bar{e} \rangle_{\tau} (\bar{v} = \bar{e}\tau \ \& \ P\tau_{\frac{\bar{x}}{a}})$ , so by the definition of Comp for procedure calls,

$\models_{M[I]} \exists \langle \bar{e} \rangle_{\tau} (\bar{v} = \bar{e}\tau \ \& \ P\tau_{\frac{\bar{x}}{a}})\{A\} \exists \langle \bar{e} \rangle_{\tau} (\bar{v} = \bar{e}\tau \ \& \ P\tau_{\frac{\bar{x}}{a}})$ , where  $p(\bar{x}:\bar{v}) \underline{\text{proc}} A$ . If  $\|\underline{\text{call}} p(\bar{x}:\bar{v})\| = k+1$ , then  $\|A\| = k$ , so

$\vdash_{H', T} \exists \langle \bar{e} \rangle_{\tau} (\bar{v} = \bar{e}\tau \ \& \ P\tau_{\frac{\bar{x}}{a}})\{A\} \exists \langle \bar{e} \rangle_{\tau} (\bar{v} = \bar{e}\tau \ \& \ P\tau_{\frac{\bar{x}}{a}})$ . Therefore, by an application of the rule for non-recursive procedure calls and lemma 7,  $\vdash_{H', T} P\{\underline{\text{call}} p(\bar{a}:\bar{e})\}Q$ .

### Appendix III. A sample proof

The 91-function (see [12]) is the function

$$f(x) = \begin{cases} x-10, & \text{if } x > 100 \\ 91, & \text{otherwise} \end{cases} \quad \text{over the set of natural numbers.}$$

We fix  $L_1$  and  $L_2$  to be languages for the natural numbers and construct a procedure in  $A1[L_1, L_2]$  to compute this function:

```
mccarthy (x:v) proc if v > 100 then x := v-10
                                     else begin new z;
                                     call mccarthy (z:v+11);
                                     call mccarthy (x:z) end.
```

We will indicate how the formulas  $v > 100 \{ \text{call } \text{mccarthy } (x:v) \} x = v-10$  and  $v \leq 100 \{ \text{call } \text{mccarthy } (x:v) \} x = 91$  may be proved. We first display a proof of a more general formula,  
 $u_1 = x \& u_2 = v \{ \text{call } \text{mccarthy } (x:v) \} u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v-10)$ .  
 Let T denote a complete proof system for the natural numbers.

Note: Implicit use of T and the rule of consequence is indicated by 'algebra'.

- (1)  $u_2 = v \& (v \leq 100 \supset v-10 = 91) \& (v > 100 \supset v-10 = v-10) \{ x := v-10 \}$   
 $u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v-10)$   

[axiom of assignment]
- (2)  $u_1 = x \& u_2 = v \& v > 100 \supset . u_2 = v \& (v \leq 100 \supset v-10 = 91) \& (v > 100 \supset v-10 = v-10)$   

[T]
- (3)  $u_1 = x \& u_2 = v \& v > 100 \{ x := v-10 \} u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v-10)$   

[(1), (2), rule of consequence]

$$(4) \quad u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v - 10) \{ \underline{\text{begin}} \quad \underline{\text{end}} \}$$

$$u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v - 10)$$

[axiom of compound  
statements]

$$(5) \quad u_1 = x \& u_2 = v \{ \underline{\text{call}} \quad r(x:v) \} u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v - 10)$$

[assumption]

$$(6) \quad u_1 = z \& u_2 = v + 11 \{ \underline{\text{call}} \quad r(z:v+11) \}$$

$$u_2 = v + 11 \& (v \leq 89 \supset z = 91) \& (v > 89 \supset z = v + 1)$$

[(5), rule of parameter  
substitution, algebra]

$$(7) \quad u_1 = z \& u_2 = v \{ \underline{\text{call}} \quad r(z:v+11) \}$$

$$u_2 = v \& (v \leq 89 \supset z = 91) \& (v > 89 \supset z = v + 1)$$

[(6), rule of variable  
substitution:  $\frac{u_2 + 11}{u_2}$ ,

algebra]

$$(8) \quad u_1 = z \& u_2 = v \& u_2 \leq 100 \{ \underline{\text{call}} \quad r(z:v+11) \}$$

$$u_2 = v \& (v \leq 89 \supset z = 91) \& (v > 89 \supset z = v + 1)$$

[T, (7), rule of consequence]

$$(9) \quad u_2 \leq 100 \{ \underline{\text{call}} \quad r(z:v+11) \} u_2 \leq 100$$

[axiom of invariance]

$$(10) \quad u_1 = z \& u_2 = v \& u_2 \leq 100 \{ \underline{\text{call}} \quad r(z:v+11) \} u_2 \leq 100$$

[T, (9), rule of consequence]

$$(11) \quad u_1 = z \& u_2 = v \& u_2 \leq 100 \{ \underline{\text{call}} \quad r(z:v+11) \}$$

$$u_2 = v \& (v \leq 89 \supset z = 91) \& (v > 89 \supset z = v + 1) \& u_2 \leq 100$$

[(8), (10), rule of  
conjunction]

$$(12) \quad u_1 = x \& u_2 = z \{ \underline{\text{call}} \ r(x:z) \} u_2 = z \& (z \leq 100 \supset x = 91) \& (z > 100 \supset x = z - 10)$$

[ (5), rule of parameter  
substitution]

$$(13) \quad u_1 = x \& u_3 = z \{ \underline{\text{call}} \ r(x:z) \}$$

$$u_3 = z \& (z \leq 100 \supset x = 91) \& (z > 100 \supset x = z - 10)$$

[ (12), rule of variable  
substitution:  $\frac{u_3}{u_2}$  ]

$$(14) \quad u_1 = x \& u_3 = z \& u_2 \leq 100 \& u_2 = v \& (v \leq 89 \supset u_3 = 91) \& (v > 89 \supset u_3 = v + 1)$$

$$\{ \underline{\text{call}} \ r(x:z) \} u_3 = z \& (z \leq 100 \supset x = 91) \& (z > 100 \supset x = z - 10)$$

[T, (13), rule of  
consequence]

$$(15) \quad u_2 \leq 100 \& u_2 = v \& (v \leq 89 \supset u_3 = 91) \& (v > 89 \supset u_3 = v + 1) \{ \underline{\text{call}} \ r(x:z) \}$$

$$u_2 \leq 100 \& u_2 = v \& (v \leq 89 \supset u_3 = 91) \& (v > 89 \supset u_3 = v + 1)$$

[axiom of invariance]

$$(16) \quad u_1 = x \& u_3 = z \& u_2 \leq 100 \& u_2 = v \& (v \leq 89 \supset u_3 = 91) \& (v > 89 \supset u_3 = v + 1)$$

$$\{ \underline{\text{call}} \ r(x:z) \} u_2 \leq 100 \& u_2 = v \& (v \leq 89 \supset u_3 = 91) \& (v > 89 \supset u_3 = v + 1)$$

[T, (15), rule of  
consequence]

$$(17) \quad u_1 = x \& u_3 = z \& u_2 \leq 100 \& u_2 = v \& (v \leq 89 \supset u_3 = 91) \& (v > 89 \supset u_3 = v + 1)$$

$$\{ \underline{\text{call}} \ r(x:z) \} u_2 \leq 100 \& u_2 = v \& (v \leq 89 \supset u_3 = 91) \&$$

$$(v > 89 \supset u_3 = v + 1) \& u_3 = z \& (z \leq 100 \supset x = 91) \& (z > 100 \supset x = z - 10)$$

[ (14), (16), rule of  
conjunction]

$$(18) \quad [u_2 \leq 100 \& u_2 = v \& (v \leq 89 \supset u_3 = 91) \& (v > 89 \supset u_3 = v+1) \&$$

$$u_3 = z \& (z \leq 100 \supset x = 91) \& (z > 100 \supset x = z-10)] \supset$$

$$[u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v-10)]$$

[T]

$$(19) \quad u_1 = x \& u_3 = z \& u_2 \leq 100 \& u_2 = v \& (v \leq 89 \supset u_3 = 91) \& (v > 89 \supset u_3 = v+1)$$

$$\{\text{call } r(x:z)\} u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v-10)$$

[(17), (18), T, rule of  
consequence]

$$(20) \quad x = x \& z = z \& u_2 \leq 100 \& u_2 = v \& (v \leq 89 \supset z = 91) \& (v > 89 \supset z = v+1)$$

$$\{\text{call } r(x:z)\} u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v-10)$$

[rule of variable

substitution:  $\frac{x, z}{u_1, u_3}$ ]

$$(21) \quad u_2 = v \& (v \leq 89 \supset z = 91) \& (v > 89 \supset z = v+1) \& u_2 \leq 100$$

$$\{\text{call } r(x:z)\} u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v-10)$$

[T, (20), rule of

consequence]

- (22)  $u_2 = v \& (v \leq 89 \supset z = 91) \& (v > 89 \supset z = v + 1) \& u_2 \leq 100$   
 $\{\underline{\text{begin}} \ \underline{\text{call}} \ r(x:z) \ \underline{\text{end}}\} u_2 = v \& (v \leq 100 \supset x = 91) \&$   
 $(v > 100 \supset x = v - 10) \quad [(4), (21), \text{rule of}$   

compound statements]

(23)  $u_1 = z \& u_2 = v \& u_2 \leq 100 \{\underline{\text{begin}} \ \underline{\text{call}} \ r(z:v+11); \ \underline{\text{call}} \ r(x:z) \ \underline{\text{end}}\}$   
 $u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v - 10)$   
 $[(11), (22), \text{rule of}$   

compound statements]

(24)  $z = z \& u_2 = v \& u_2 \leq 100 \{\underline{\text{begin}} \ \underline{\text{call}} \ r(z:v+11); \ \underline{\text{call}} \ r(x:z) \ \underline{\text{end}}\}$   
 $u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v - 10)$   
 $[(23), \text{rule of variable}$   

substitution:  $\frac{z}{u_1}]$

(25)  $u_1 = x \& u_2 = v \& v \leq 100 \& z = e_0$   
 $\{\underline{\text{begin}} \ \underline{\text{call}} \ r(z:v+11); \ \underline{\text{call}} \ r(x:z) \ \underline{\text{end}}\}$   
 $u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v - 10)$   
 $[(24), T, \text{rule of}$   

consequence]

(26)  $u_1 = x \& u_2 = v \& v \leq 100$   
 $\{\underline{\text{begin}} \ \underline{\text{new}} \ z; \ \underline{\text{call}} \ r(z:v+11); \ \underline{\text{call}} \ r(x:z) \ \underline{\text{end}}\}$   
 $u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v - 10)$   
 $[(25), \text{rule of variable}$   

declarations]

(27)  $u_1 = x \& u_2 = v \{\underline{\text{if}} \ v > 100 \ \underline{\text{then}} \ x := v - 10$   
 $\quad \underline{\text{else}} \ \underline{\text{begin}} \ \underline{\text{new}} \ z;$   
 $\quad \quad \underline{\text{call}} \ r(z:v+11);$   
 $\quad \quad \underline{\text{call}} \ r(x:z) \ \underline{\text{end}}\}$   
 $u_2 = v \& (v \leq 100 \supset x = 91) \& (v > 100 \supset x = v - 10)$   
 $[(3), (26), \text{rule of}$   

conditional statements]





THE UNIVERSITY OF CHICAGO PRESS

CHICAGO, ILLINOIS

1900

THE UNIVERSITY OF CHICAGO PRESS

CHICAGO, ILLINOIS

1900

THE UNIVERSITY OF CHICAGO PRESS

CHICAGO, ILLINOIS

1900

THE UNIVERSITY OF CHICAGO PRESS

CHICAGO, ILLINOIS

1900

THE UNIVERSITY OF CHICAGO PRESS

CHICAGO, ILLINOIS

1900

THE UNIVERSITY OF CHICAGO PRESS

CHICAGO, ILLINOIS

1900

THE UNIVERSITY OF CHICAGO PRESS

CHICAGO, ILLINOIS

1900

THE UNIVERSITY OF CHICAGO PRESS

CHICAGO, ILLINOIS

1900

THE UNIVERSITY OF CHICAGO PRESS

CHICAGO, ILLINOIS

1900

THE UNIVERSITY OF CHICAGO PRESS

## References

1. Hoare, C.A.R. An axiomatic basis for computer programming.  
Comm. ACM 12, 10 (Oct. 1969), 576-580.
2. Hoare, C.A.R. Procedures and parameters: an axiomatic approach. In Symposium on Semantics of Algorithmic Languages, E. Engeler, Ed., Springer-Verlag, Berlin, 1971, pp. 102-116.
3. Floyd, R.W. Assigning meanings to programs. Proc. Amer. Math. Soc. Symposia in Applied Mathematics 19 (1967), 19-31.
4. Lauer, P.E. Consistent formal theories of the semantics of programming languages. Tech. Rep. TR.25.121, IBM Laboratory, Vienna, 1971.
5. Cook, S. Axiomatic and interpretive semantics for an Algol fragment. Lecture notes, to appear as Tech. Rep., Department of Computer Science, University of Toronto.
6. Hoare, C.A.R., Lauer, P.E. Consistent and complementary formal theories of the semantics of programming languages.  
Acta Infor. 3 (1974), 135-153.
7. Igarashi, S., London, R.L., Luckham, D.C. Automatic program verification I: a logical basis and its implementation. AIM-200, Artificial Intelligence Laboratory, Stanford U., July, 1973.
8. Clint, M., Hoare, C.A.R. Program proving: jumps and functions.  
Acta Infor. 1 (1971), 214-224.
9. Hoare, C.A.R. Proof of correctness of data representations.  
Acta Infor. 1 (1972), 271-281.
10. Cook, S., Oppen, D. An assertion language for data structures.  
Proc. of Second ACM Symposium on Principles of Programming

Languages, 1975 (to appear).

11. Oppen, D. On logic and program verification. Ph.D. Th., Department of Computer Science, University of Toronto (to appear).
12. Manna, Z., Ness, S., Vuillemin, J. Inductive methods for proving properties of programs. Comm. ACM 16 (Aug. 1973), 491-502.

TECHNICAL REPORTS  
Department of Computer Science  
September 68      -      June 74

<u>Number</u>	<u>Author</u>	<u>Title</u>
* 1	T.E. Hull	The Numerical Integration of Ordinary Differential Equations
* 2	J.C. Mason	Chebyshev Methods for Separable Partial Differential Equations
* 3	Neil Frederick Stewart	The Comparison of Numerical Methods for Ordinary Differential Equations
* 4	J.N.P. Hume and C.B. Rolfson	Scheduling for Fast Turnaround in Job-At-A-Time Processing
* 5	G.F. Gabel	A Predictor-Corrector Method Using Divided Differences
* 6	Chandler Davis and W.M. Kahan	The Rotation of Eigenvectors by a Perturbation - III
*7	D.J. Cohen and C.C. Gotlieb	The Syntax Graph: A List Structure for Representing Grammars
* 8	D. Tsichritzis	Measures on Countable Sets
9	Brian Smith	Error Bounds, Based Upon Gerschgorin's Theorems, for the Zeros of a Polynomial
*10	M. Puzin	Simulation of Cellular Patterns by Computer Graphics
*11	J.C. Mason	Orthogonal Polynomial Approximation Methods in Numerical Analysis
*12	Donald M. Kaplan	Recursion Induction Applied to Generalized Flowcharts

---

\* Not available

## TECHNICAL REPORTS

(continued)

<u>Number</u>	<u>Author</u>	<u>Title</u>
*13	Sunil K. Pal	Numerical Solution of First-Order Hyperbolic Systems of Partial Differential Equations
*14	Pat Conroy	Simulation of Texture by Computer Graphics
*15	J.C. Mason and I. Barrodale	Two Simple Algorithms for Discrete Rational Approximation
*16	S. Gerschgorin and Paul Turan	Two Translations in Numerical Analysis
*17	George Olshevsky Jr.	An Automated Aid to Visualizing Convex Hypersolids
*18 (2nd ed.)	Derek Gordon Corneil	Graph Isomorphism
*19	Alan Borodin	Computational Complexity and the Existence of Complexity Gaps
*20	John David Duffin	A Language for Line Drawing
*21	A. Ballard	Transformations on Programs
*22	S.A. Cook	Linear Time Simulation of Deterministic Two-Way Pushdown Automata
* 23 (2nd ed.)	D. Tsichritzis (Editor)	Topics in Operating Systems
*24 (2nd ed.)	R. Bunt and D. Tsichritzis	A Selective Annotated Bibliography for Operating Systems
*25	J.C. Mason and I. Farkas	Continuous Methods for Free Boundary Problems

---

\* Not available

TECHNICAL REPORTS  
(continued)

<u>Number</u>	<u>Author</u>	<u>Title</u>
*26	W.A. Walker and C.C. Gotlieb	A Top Down Algorithm for Constructing Nearly-Optimal Lexographic Trees
*27	D. Tsichritzis	Iff Programs
*28	L.W. Jackson	Automatic Error Analysis for the Solution of Ordinary Differential Equations
*29	T.E. Hull W.H. Enright B.M. Fellen and A.E. Sedgwick	Comparing Numerical Methods for Ordinary Differential Equations
*30	Stanley Cabay	Second-Order Hyperbolic Equations with Data on Two Intersecting Boundaries
*31	Douglas Dale Olesky	Inclusion Regions for Partitioned Matrices
*32	J. Ian Munro	Some Results in the Study of Algorithms
*33	D. Tsichritzis	Modular System Description
*34	J. Mylopoulos	On the Relation of Graph Automata and Graph Grammars
*35	P.H. Roosen-Runge	The Algebra of Distributionally Defined Classifications
*36	T.E. Hull  T.E. Hull W.H. Enright and A.E. Sedgwick	Proving the Correctness of Algorithms  The Correctness of Numerical Algorithms

---

\* Not available

TECHNICAL REPORTS

(continued)

<u>Number</u>	<u>Author</u>	<u>Title</u>
37	Erold W. Hinds	Square Roots of an Orthogonal Matrix
*38	D. Tsichritzis	Topics in Operating Systems Revisited
39	David G. Kirkpatrick	On the Additions Necessary to Compute Certain Functions
*40	P. Keast	Multi-Dimensional Quadrature Formulae
*41	Gordon D. Mulligan	Algorithms for Finding Cliques of a Graph
42	Stephen A. Cook and Robert A. Reckhow	Diagonal Theorems for Random Access Machines
*43	C.J.C. Lee	Translation of Context-Free Programming Languages Using Semantic Trees
*44	D. Tsichritzis	Lecture Notes on Operating Systems
*45	David Blair Coldrick	Methods for the Numerical Solution of Integral Equations of the Second Kind
*46	Wayne Enright	Studies in the Numerical Solution of Stiff Ordinary Differential Equations
*47	Barry Graham	An Algorithm to Determine the Chromatic Number of a Graph
*48	Philip A. Bernstein	Description Problems in the Modeling of Asynchronous Computer Systems
*49	Rudolf Mathon	On the Approximation of Elliptic Boundary Value Problems by Fundamental Solutions

---

\* Not available



TECHNICAL REPORTS  
(continued)

<u>Number</u>	<u>Author</u>	<u>Title</u>
50	George Tourlakis	Some Results in Computational Topology
51	R. Michael Wharton	Grammatical Inference and Approximation
*52	John Mylopoulos Norm Badler Lou Melli and Nicholas Roussopoulos	An Introduction to 1.pak, A Programming Language for AI Applications
53	Arthur Sedgwick	An Effective Variable Order Variable Step Adams Method
*54	Frank Wm. Tompa and C.C. Gotlieb	Choosing a Storage Schema
55	John Mylopoulos Norman Badler Walter Berndt and Lucio Melli	The 1.pak Reference Manual
56	Keith O. Geddes	Algorithms for Analytic Approximation
57	Robert Moenck	Studies in Fast Algebraic Algorithms
58	Daniel Brand	Resolution and Equality in Theorem Proving
*59	R.D. Tennent	Mathematical Semantics and Design of Programming Languages
60	G.Hall W.H. Enright T.E. Hull and A.E. Sedgwick	DETEST: A Program for Comparing Numerical Methods for Ordinary Differential Equations

---

\* Not available

# TECHNICAL REPORTS

(continued)

<u>Number</u>	<u>Author</u>	<u>Title</u>
*61	C.C. Gotlieb	Data Types and Structures, A Synthetic Approach
62	C.A. Steele A.E. Sedgwick	DEFT: A Disciplined Extension of Fortran
*63	T.E. Hull J.J. Hofbauer	Language Facilities for Multiple Precision Floating Point Computation, with Examples and the Description of a Preprocessor
64	Philip R. Cohen	A Prototype Natural Language Understanding System
*65	D.G. Corneil	The Analysis of Graph Theoretical Algorithms
66	T.E. Hull W.H. Enright	A Structure for Programs that Solve Ordinary Differential Equations
67	Bengt Lindberg	Optimal Stepsize Sequences and Requirements for the Local Error for Methods for (Stiff) Differential Equations
68	W.H. Enright R. Bedet I. Farkas T.E. Hull	Test Results on Initial Value Methods for Non-Stiff Ordinary Differential Equations
69	W.H. Enright T.E. Hull B. Lindberg	Comparing Numerical Methods for Stiff Systems of Ordinary Differential Equations
70	C.C. Gotlieb A.L. Furtado	Data Schemata Based on Directed Graphs
71	T.L. de Carvalho	Some Results in Automatic Theorem- Proving with Applications in Elementary Set Theory and Topology.
72	Yiu-Sang Moon	Some Numerical Experiments on Number Theoretic Methods in the Approximation of Multi-dimensional Integrals.

TECHNICAL REPORTS  
(continued)

<u>Number</u>	<u>Author</u>	<u>Title</u>
73	Lucio F. Melli	The 2.pak Language: Primitives for AI Applications.
74	David G. Kirkpatrick	Topics in the Complexity of Combinatorial Algorithms.
75	Gerald A. Gorelick	A Complete Axiomatic System for Proving Assertions about Recursive and Non-Recursive Programs